THE DEVELOPMENT OF A FINITE ELEMENT ALGORITHM FOR BLAST MODELLING

Work on this project was conducted under the auspices of the
Canada Centre for Mineral and Energy Technology,
Energy, Mines and Resources Canada

by

D. Gordon Dunbar & Associates Ltd.
5370 Keith St.
Burnaby, B. C.
V5J 3C5
(604) 430-8359/430-5362

DSS Contract Serial No. 23440-7-9179/01-SQ
DSS Contract File No. 33SQ.23440-7-9179
Scientific Authority: Dr. Somchet Vongpaisal

88 pp

March 1988

MICROMEDIA

EXECUTIVE SUMMARY

Blast design is largely an empirical procedure by which the three main concerns of a blast designer, namely breakage, fragmentation, and damage, are estimated using data from full scale instrumented tests, equations derived from such tests, and judgement based on experience. Although empirical methods are generally successful, they rely to a considerable extent on the gradual process of learning how blasting affects rock in a particular mine. For example, rock fragmentation can vary considerably depending on such factors as pre-existing rock structure, rock type, rock strength, blast size and configuration, etc. It is difficult to estimate the effect of changes in these parameters without a lot of experimentation.

The alternative to an empirical approach is an analytical or numerical model which would predict the effect of a blast. Recently significant advances have been made in the development of mathematical models to describe fragmentation processes. Thus it is now possible to develop analytical or numerical methods which do take into account factors known to affect breakage, fragmentation and damage.

The work reported herein represents the first step toward the development of a computational model of blasting which incorporates some of these recent advances and which is implemented on a microcomputer. A complete blast modelling program together with a user's manual and technical documentation are provided. Although the use of such a model in routine blast design is still a distant goal, it is believed that it could be used as an aid to judgement or as a means of understanding field measurements. In this sense, it could bring a more quantitative basis to blast design.

SOMMAIRE EXÉCUTIF

Le dessein de sautage est en grande partie une procédure empirique où les trois intérêts majeurs d'un ingénieur de sautage, c'est-à-dire la rupture de la roche, la fragmentation, et le dommage, sont estimeés faisant usage des données acquises au cours de tests exécutés grâce à une batterie complète d'instruments, d'équations dérivés au cours de tests semblables, et de jugement fondé sur l'experience. Quoique les méthodes empiriques donnent généralement de bons résultats, elles dépendent essentiellement sur l'apprentissage graduel de la manière dont le sautage affecte la roche dans chaque mine en particulier. Par exemple, la fragmentation de la roche varie considérablement selon des facteurs tels que: la structure de roche pré-existante, le type de roche, la résistance de cette dernière, la dimension et la configuration de la détonation, et caetera. Il est difficile d'éstimer l'effet de changements en ces paramètres sans beaucoup d'expérimentation.

L'alternatif à une approche empirique est un modèle analytique ou numerique qui saurait prédire l'effet d'une explosion. Récement des progrès significatifs ont été faits dans le développement de modèles mathematiques capable de décrire le procédé de fragmentation de la roche. Ainsi, il est désormais possible de développer méthodes analytiques ou numeriques qui tient compte des facteurs affectant la rupture, la fragmentation et le dommage.

L'ouvrage qui est exposé ici représente le premier pas vers le développement d'un modèle a comptes qu'incorpore quelques-uns des récents progrès qui onts étés marqués dans ce domaine. Ci-joint on trouvera un programme d'ordinateur, un manuel pour l'usager, et une documentation qui traite du sujet. Quoique l'usage d'un tel outil de travail à des fins de sautage ne soit encore qu'un but éloigné, il pourra servir comme aide au jugement ou comme un moyen afin de faciliter la compréhension des données des travaux pratiques. Dans ce sens, cette technique offre une base plus quantitative au dessein de sautage.

TECHNICAL ABSTRACT

Although empirical methods of blast design are generally successful, they still rely on a gradual process of learning how blasting affects rock in a particular mine, either through full scale instrumented tests or judgement based on experience. A computational model, if suitably formulated, would be able to assist this process by predicting rock behaviour due to a blast, thus leading to more reliable blast design.

Computational blast models can be divided into near field and far field models. Near field models attempt to simulate the nonelastic behaviour which occurs near the blasthole and are useful for determining the level of comminution caused by a blast. Far field models ignore the material behaviour near the blasthole and are used to predict particle velocities or to study elastic wave propagation effects. Geometrically, the division between near and far field models is a radius known as the radius of the equivalent cavity. At distances greater than this radius the rock is assumed to behave elastically .

A complete two dimensional finite element wave propagation program is developed. Significant features of this program include a silent or absorbing boundary which eliminates undesirable reflections from artificial external boundaries of the finite element model, a restart facility, and a simple tensile failure fragmentation model. An explicit time integration scheme using lumped matrices is used which avoids the storage of large matrices, thus enabling quite large models to be set up on a microcomputer. The program is part of a package which includes a finite element mesh generator, a mesh display program, and output display programs. The package is completely documented and a user's manual is provided.

Examples of the use of the finite element program for both near and far field models are given. The program was also used to predict peak velocities due to a single large blast in a currently operating. Reasonable agreement was found despite the simplifying assumptions made.

It is felt that this work represents a first step toward the development of blast models which could be used as an aid to understanding and improving blast design.

ACKNOWLEDGEMENTS

The work described in this report could not have been done without the help and support of Dr. Somchet Vongpaisal of the Mining Research Laboratories, Canada Centre for Mineral and Energy Technology.

The project was funded by the Unsolicited Proposals Program of the Department of Supply and Services, Canada.

Dr. T. R. Yu, Chief Rock Mechanics and Research Engineer, Kidd Creek Mines Ltd., provided information for a case study.

Mr. G. Hodge did an excellent job of drawing the figures.

TABLE OF CONTENTS

Page

Executive Summary 1
Sommaire Exécutif ii
Technical Abstract iii
Acknowledgements iv

CHAPTER 1 - Introduction

1.1 Blast Design 1
1.2 Computational Models of Blasting 2
1.3 Scope of Report 4
1.4 Notation 5

CHAPTER 2 - Finite Element Model

2.1 Finite Element Approximation 7
2.2 Equations of Motion 9
2.3 Mass and Damping Matrices, Internal Force Vector 11
2.4 Load Vector 12
2.5 Explicit Solution Scheme 13
2.6 Constitutive Model for Fragmentation 14
2.7 Silent Boundary 17
2.8 Determination of the Equivalent Cavity 19

CHAPTER 3 - Applications

3.1 One Dimensional Test Problems 21
3.2 Two Dimensional Examples 25
3.3 Kidd Creek Mine Case Study 34

CHAPTER 4 - Conclusions and Recommendations

4.1 Conclusions 41
4.2 Recommendations 42

REFERENCES 43

APPENDICES

A - Elastic Constitutive Model 47
B - User's Manual 49
C - Program Listing 61

CHAPTER 1

Introduction

1.1 Blast Design

The concerns of a blast designer may be divided into three parts:

1.) breakage - what is the amount of explosive required to break a given burden of rock?
2.) fragmentation - will the rock break into pieces that can be handled by existing equipment?
3.) damage - will the blast result in permanent disturbance (or destruction) of remaining material?

All of these are inter-related and the final design usually involves some kind of compromise between each of the concerns.

There exist simple analytical and semi-numerical models to describe each of the above design concerns (Sharpe, 1942; Duvall, 1953; Starfield and Pugliese, 1968; Harries, 1977) In addition, as a result of many full scale instrumented experiments, good qualitative descriptions of the blast process are available (see, for example, Brady and Brown, 1985, pp. 446-451). In most cases, the models have successfully described the blast process as seen in experiments, but fail to be a reliable predictive tool for design. There are many reasons for this, one of which is that the models assume that the rock is initially intact, homogeneous and isotropic. This contradicts experimental observations which suggest that breakage, fragmentation and damage are all controlled by pre-existing geological structure. This and other valid objections to simplified blast models are discussed in a recent report by INCO Ltd. (1987).

Probably the most successful blast design procedures are empirical in nature. For example, one of the more highly developed of these is the Swedish blasting method (Langefars and Kihlstrom, 1978). The basis of this method is an empirically derived relationship between the amount of explosive required to break a given burden and parameters such as rock strength and blasthole spacing, which are either easily measured or well constrained.

Empirical methods also exist to predict fragmentation sizes. Kuznetsov (1973) developed an equation which relates the mean fragment size to blast energy, blasthole size and rock strength. The size distribution of fragments in blasted can be described by an equation developed by Rosin and Rammler (1933) for powdered coal. Cunningham (1983) combined the Kuznetsov and Rosin-Rammler equations to give a means of estimating mean fragment and size distribution based on the blast geometry.

One criticism of these equations, mentionned previously, is that they do not take into account the pre-existing jointing or fractures. For example, if the blasthole spacing is greater than the average fracture spacing, then the size distribution of the fragments will tend to be governed by the pre-existing structure. Some preliminary work has been done to incorporate the effects of

pre-existing structure (see report by INCO Ltd., 1987)

Damage may also be described by simple models or empirical methods. Again, the simple models do not take pre-existing structure into account. Empirical methods usually relate peak particle velocity, which is measurable, to qualitative descriptions of rock damage. Two problems exist with this approach:

- Peak particle velocities can vary by an order of magnitude or more depending on rock type, rock strength, blast size and configuration, detonation characteristics etc. It is difficult to estimate the effect of changes in these parameters without a lot of experimentation.
- Peak particle velocities cannot be measured close to the blast which is the area of interest to the blast designer. Extrapolation from measurements made in the 'far field' is sometimes used to predict velocities.

In addition, the definition of damage may change depending on the application.

It is evident that the state of practice of blast design is largely empirical and relies to a considerable extent on the gradual process of learning how blasting affects rock in a particular mine. The alternative to empiricism is, of course, an analytical or numerical model which, in a physical manner, accounts for factors known to affect rock fragmentation such as pre-existing geological structure, rock inhomogeneity and detonation characteristics. However, the goal of developing such a model remains elusive mainly because of the poor understanding of the above factors and also because of the sheer complexity of describing the above factors in a form suitable for an analytical or numerical model.

Nevertheless, much work has been done on developing computational schemes to describe some aspects of the blasting process. The goal of this work is to provide a more rational basis for design. The next section reviews some of this work.

1.2 Computational Models of Blasting

A qualitative model of rock blasting divides the region surrounding the blast into three zones:

- an inner zone of radius r_a in which the blast pressure near the blasthole exceeds the compressive strength of the rock and decays to equal the compressive strength of the rock at $r = r_a$
- an annular zone $r_a \leq r \leq r_1$ in which the stress is less than the compressive strength but greater than the tensile strength
- an exterior region $r > r_1$ in which the rock is assumed to behave elastically.

The boundary $r = r_1$ defines the 'equivalent cavity', a concept introduced by Sharpe (1942). Pressures in and near the blasthole can be as large as 10^{10} Pa so that rock in the inner zone $r \leq r_a$ is completely crushed. Radial cracks may occur in the annulus $r_a \leq r \leq r_1$.

As will be discussed, computational models of blasting attempt to model the rock behaviour inside or outside the equivalent cavity. The blast process is modelled as a transient wave propagation problem. The spatial derivatives of the equations of motion are approximated by finite differences or

finite elements. Time derivatives are approximated by finite differences which permits the solution (displacements, velocities and accelerations) to be computed at successive time steps. Variations in material properties can be incorporated in the approximation for spatial derivatives.

Development of such models has been motivated by a number of problems. For many years classified work on the effects of nuclear blasts on missile silos has been done under the auspices of the U. S. Defence Department Constantino (1967) and Constantino *et al* (1967) modelled elastic wave propagation from a surface crater formed by a nuclear blast. Heller (1972) used a finite element model to determine the response of a large dam to a deep underground nuclear explosion. Smith (1975) and Bolt and Smith (1976) developed a finite element model for shock wave propagation to be used as an aid to data interpretation in exploration seismology.

The behaviour of the rock within the equivalent cavity is obviously nonelastic. Various attempts have been made to develop computational models for blasting within the equivalent cavity (the near field). Kutter (1967), Porter (1970) and Bhandari (1976) used one dimensional finite element models, with pressures exceeding the compressive strength of the rock, to investigate near field behaviour. A similar approach was taken by Santich (1976) except that viscoelastic material properties were incorporated in an attempt to model the nonlinear behaviour in the near field. Unfortunately none of these models was very successful. In addition, one dimensional models cannot take the geometrical spreading of stress waves into account. Ingraffea (1977) used static finite element analysis with incremental loading to study fracture propagation in the near field.

The large levels of comminution required for the in situ extraction of oil from oil shale prompted many studies of the effect of blasting on fragmentation. Grady and Kipp (1980) developed a constitutive model of fragmentation which is based on the activation and growth of an initial distribution of cracks. A damage parameter, ranging between zero and one, describes the level of crack growth and the decrease in modulus as cracking proceeds. Similar models were developed by Dienes and Margolin (1980) and incorporated into computer programs by Margolin (1981). Digby et al (1985) compared two constitutive models of blasting: the one developed by Grady and Kipp (1980) and a more complicated one developed by Nilsson and Oldenburg (1983). One of the more advanced near field blast models published to date is that by Taylor *et al* (1986) who used a damage accumulation model in a finite element code to predict comminution surrounding crater blasts in oil shale. The predictions were corroborated by field experiments and analysis of drill cores surrounding the craters.

Two dimensional dynamic finite element models have been developed to model the effects of blasting at larger distances from the blast (the far field). Shipley *et al* (1967) computed elastic fields resulting from a cylindrical charge using an axisymmetric finite element model. Verner (1971) proposed a similar two dimensional model. More recently, Blair (1981) used static and dynamic finite element analysis to model ground surface displacements due to a cylindrical charge. Shastri and Maiumdar (1982) developed a three dimensional dynamic finite element code and used it to model a bench blast. Several finite difference programs have also been developed to model wave propagation and have been used for blasting applications. (Heusinkveld *et al*, 1975; Sweagle, 1978).

Evidently, the development of a computational blast model can proceed in two ways depending

on the concerns of the blast designer. If it is necessary to determine the level of comminution, a near field equivalent cavity model would be appropriate. However, if one is interested in potential damage or ground motion during blasting, a far field elastic model of the region lying outside the equivalent cavity would be useful for predicting particle velocities.

These two problems form a natural division since the high frequencies associated with the blast pulse are rapidly attenuated within the equivalent cavity (the earth acts like a low pass filter). Such high frequencies require very fine spatial discretization in a computational model. Outside the equivalent cavity, where frequencies are lower, coarser discretization is possible and thus larger models are practicable. In addition, in the far field the response is relatively insensitive to the details of the blast pulse so that it is unnecessary to model such details.

It is believed that the derivation of constitutive models to describe the near field fragmentation process represents a significant advance in the development of a computational model of blasting. It is now possible to develop models which do take into account the pre-existing structure and its effect on breakage, fragmentation and damage. However, it must be emphasized that the use of such models in routine blast design is still a distant goal. Rather, it is more likely that computational models will be used as an aid to judgement or as a means of understanding field measurements such as peak particle velocity.

1.3 Scope of Report

This report will concentrate on a description of the theory and computational aspects of transient finite element models for both the near and far field. Given the time history of the pressure pulse in the blasthole, the qualitative description of blasting given in the last section is sufficient to enable the development of a relatively simple method for determining the radius of the equivalent cavity and the pressure time history on the walls of that cavity. This is used as input to an elastic finite element wave propagation model. For the near field, constitutive models of fragmentation, or other material models, must be used. These can be incorporated rather easily into a transient finite element model.

The methodology will be illustrated by means of a fully documented set of computer programs written in FORTRAN77. The programs incorporate advanced input and output processors, with graphics capability, as well as many recently developed techniques of dynamic finite element analysis. Most analyses are within the storage capability of an IBM or compatible microcomputer. Examples of the use of this program will be given.

1.4 Notation

Bold face characters will denote vector, tensor or matrix quantities. Subscripted characters in plain face will, in general, denote components of vectors, tensors or matrices. Subscripted Greek symbols will usually refer to quantities associated with a material medium. A dot (e.g., $\dot{\mathbf{u}}$) denotes the time derivative and a hat (e.g., $\hat{\mathbf{x}}$) denotes an interpolated quantity. A superscript T denotes the transpose of a matrix or vector. SI units will be used as given in the Canadian Metric Practice Guide published by the Canadian Standards Association (1976).

The more commonly used symbols are given below.

$\mathbf{b}$ - vector of body forces $\mathbf{b} = (b_x, b_y)^T$
B - bulk modulus (pascals)
$\mathbf{B}$ - strain interpolation matrix (m^{-1})
$\mathbf{C}$ - damping matrix (m/s/newton)
$\mathbf{D}$ - elasticity matrix
$\mathbf{e}$ - vector of strains in element $\mathbf{e} = (\epsilon_{xx}, \epsilon_{yy}, \gamma_{xy})^T$ where $\gamma_{xy} = 2\epsilon_{xy}$
E - Young's modulus (pascals)
$\mathbf{f}$ - nodal forces (newtons)
G - shear modulus (pascals)
h - interpolation function
$\mathbf{H}$ - displacement interpolation matrix
$\mathbf{K}$ - stiffness matrix (newton/m)
$\mathbf{M}$ - mass matrix (kg)
$\mathbf{s}$ - vector of stresses in element $\mathbf{s} = (\sigma_{xx}, \sigma_{yy}, \sigma_{xy})^T$
t - time (s)
$\mathbf{u}$ - vector of nodal displacements (m)
c_p - P wave velocity (m/s)
c_s - S wave velocity (m/s)
Δt - time step (s)
δ_{ij} - Kronecker delta, $\delta_{ij} = 1$ if $i = j$, 0 otherwise
ϵ - hydrostatic strain
λ - Lame's constant (pascals)
ρ - density (kg/m^3)
σ - hydrostatic stress (pascals)
ν - Poisson's ratio

CHAPTER 2

Finite Element Model

2.1 Finite Element Approximation

Dynamic finite element analysis begins by dividing the solid into subregions (elements) defined by a number of nodes. Displacements, velocities, and accelerations (fields) are computed at the nodes by the solution procedure. Field values at other points in an element are then computed by interpolation of the values at the nodes of the element. In this way, a problem with an infinite number of degrees of freedom is converted to one with a finite number.

For example, in the element shown in Figure 2.1, a bilinear interpolation of displacements is given by

$$\hat{u}_x = \sum_{i=1}^{4} h_i u_{xi} \qquad \hat{u}_y = \sum_{i=1}^{4} h_i u_{yi}$$

where the h_i are interpolation functions given by

$$h_1 = \frac{1}{4}(1+r)(1+s) \qquad h_2 = \frac{1}{4}(1-r)(1+s)$$

$$h_3 = \frac{1}{4}(1-r)(1-s) \qquad h_4 = \frac{1}{4}(1+r)(1-s)$$

$$-1 \le r, s \le 1$$

An important property of the interpolation functions is that $h_i(r_i, s_i) = 1$.

In matrix-vector notation

$$\hat{\mathbf{u}} = \begin{pmatrix} \hat{u}_x \\ \hat{u}_y \end{pmatrix} = \begin{pmatrix} h_1 & 0 & h_2 & 0 & h_3 & 0 & h_4 & 0 \\ 0 & h_1 & 0 & h_2 & 0 & h_3 & 0 & h_4 \end{pmatrix} \begin{pmatrix} u_{x1} \\ u_{y1} \\ u_{x2} \\ u_{y2} \\ u_{x3} \\ u_{y3} \\ u_{x4} \\ u_{y4} \end{pmatrix}$$

or

$$\hat{\mathbf{u}} = [\mathbf{H}_1, \mathbf{H}_2, \mathbf{H}_3, \mathbf{H}_4] = \mathbf{H}\mathbf{u} \tag{2.1}$$

where $\mathbf{H}_i$ is the 2 by 2 submatrix

$$\mathbf{H}_i = \begin{pmatrix} h_i & 0 \\ 0 & h_i \end{pmatrix}$$

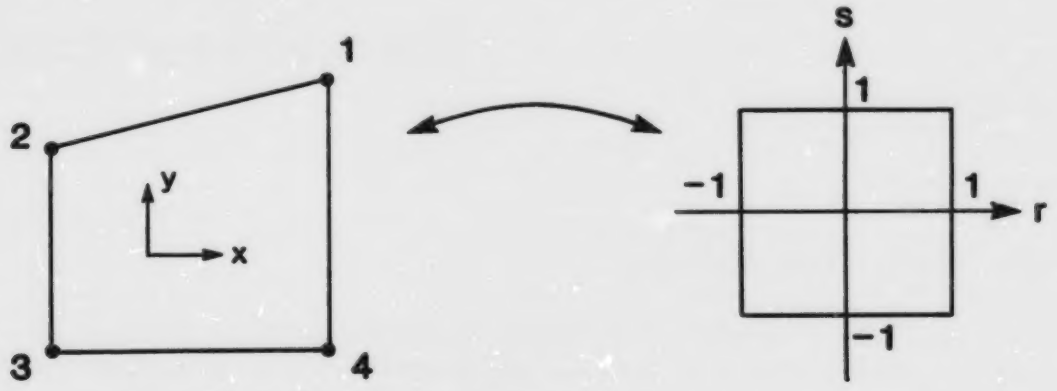


Figure 2.1

Linear isoparametric element

This approximation will be used in all subsequent work. Note that the coordinates x and y can be interpolated by exactly the same functions

$$x = \sum_{i=1}^{4} h_i x_i \qquad y = \sum_{i=1}^{4} h_i y_i \tag{2.2}$$

Since the coordinates and field quantities are described by the same equations, the approximation is termed isoparametric (Bathe, 1982).

The strains in the element corresponding to the displacements $\hat{u}$ are given by

$$\mathbf{e} = \begin{pmatrix} \epsilon_{xx} \\ \epsilon_{yy} \\ \gamma_{xy} \end{pmatrix} = \begin{pmatrix} \dfrac{\partial \hat{u}_x}{\partial x} \\ \dfrac{\partial \hat{u}_y}{\partial y} \\ \dfrac{\partial \hat{u}_x}{\partial y} + \dfrac{\partial \hat{u}_y}{\partial x} \end{pmatrix}$$

By the chain rule

$$\begin{pmatrix} \dfrac{\partial}{\partial r} \\ \dfrac{\partial}{\partial s} \end{pmatrix} = \begin{pmatrix} \dfrac{\partial x}{\partial r} & \dfrac{\partial y}{\partial r} \\ \dfrac{\partial x}{\partial s} & \dfrac{\partial y}{\partial s} \end{pmatrix} \begin{pmatrix} \dfrac{\partial}{\partial x} \\ \dfrac{\partial}{\partial y} \end{pmatrix} \tag{2.3}$$

where the matrix of derivatives, $\mathbf{J}$, is known as the Jacobian and can be evaluated for a given r and s using Equation 2.2. Inversion of $\mathbf{J}$ allows derivatives with respect to x and y to be determined in terms of derivatives with respect to r and s. The strains can therefore be written in terms of derivatives of the interpolation functions as

$$\mathbf{e} = [\mathbf{B}_1, \mathbf{B}_2, \mathbf{B}_3, \mathbf{B}_4]\mathbf{u} = \mathbf{B}\mathbf{u} \tag{2.4}$$

where $\mathbf{B}_i$ is the 3 by 2 submatrix

$$\mathbf{B}_i = \begin{pmatrix} \frac{\partial h_i}{\partial x} & 0 \\ 0 & \frac{\partial h_i}{\partial y} \\ \frac{\partial h_i}{\partial y} & \frac{\partial h_i}{\partial x} \end{pmatrix}$$

Stresses are given by Hooke's law written in terms of the strains e and, using Equation 2.4, the nodal displacements u

$$\mathbf{s} = \begin{pmatrix} \sigma_{xx} \\ \sigma_{yy} \\ \sigma_{xy} \end{pmatrix} = \mathbf{D}\mathbf{e} = \mathbf{D}\mathbf{B}\mathbf{u} \tag{2.5}$$

where, for plane strain, the elasticity matrix D is

$$\mathbf{D} = \frac{E(1-\nu)}{(1+\nu)(1-2\nu)} \begin{pmatrix} 1 & \frac{\nu}{(1-\nu)} & 0 \\ \frac{\nu}{(1-\nu)} & 1 & 0 \\ 0 & 0 & \frac{(1-2\nu)}{2(1-\nu)} \end{pmatrix}$$

2.2 Equations of Motion

Using the approximations developed in the previous section and a basic principle of mechanics, differential equations of motion for the discretized medium may be derived.

Let the vector $\mathbf{t} = (t_x, t_y)^T$ denote the external surface forces (tractions) applied on the surface S of an element occupying a region V. Also, let the vector $\mathbf{b} = (b_x, b_y)^T$ denote distributed body forces (i.e., body forces per unit volume) acting in the element. The principle of virtual work states that the internal work done by the virtual (or arbitrary, except that they satisfy any displacement boundary conditions) displacements $\delta\mathbf{v}$ at the nodes of the element must be equal to the external work done by the tractions and body forces acting through the same virtual displacements. The external work is given by

$$\int_S \delta\hat{\mathbf{v}}^T \mathbf{t}\, dS + \int_V \delta\hat{\mathbf{v}}^T \mathbf{b}\, dV = \delta\mathbf{v}^T \int_S \mathbf{H}^T \mathbf{t}\, dS + \delta\mathbf{v}^T \int_V \mathbf{H}^T \mathbf{b}\, dV$$

where $\delta\hat{\mathbf{v}} = \mathbf{H}\delta\mathbf{v}$ from Equation 2.1. The internal work done by the stresses s acting through the strains e caused by the displacements $\delta\mathbf{v}$ is

$$\int_V \delta \mathbf{e}^T \boldsymbol{\sigma} dV = \delta \mathbf{v}^T \int_V \mathbf{B}^T \boldsymbol{\sigma} dV = \delta \mathbf{v}^T \mathbf{q} \tag{2.6}$$

where from Equation 2.4 $\delta \mathbf{e} = \mathbf{B}\delta \mathbf{v}$. The vector $\mathbf{q}$ is known as the internal force vector. Equating the external work and internal work in the element results in

$$\delta \mathbf{v}^T \mathbf{q} = \delta \mathbf{v}^T \left[\int_S \mathbf{H}^T \mathbf{t} dS + \int_V \mathbf{H}^T \mathbf{b} dV \right] \tag{2.7}$$

Since this relationship is valid for any virtual displacement, the following must hold:

$$\mathbf{q} = \int_S \mathbf{H}^T \mathbf{t} dS + \int_V \mathbf{H}^T \mathbf{b} dV \tag{2.8}$$

When dynamic loads are present, inertial forces must be taken into account. The inertial force caused by an acceleration $\ddot{\mathbf{u}}$ opposes the external dynamic loads and is given by

$$-\rho \ddot{\mathbf{u}}$$

Viscous damping forces may also be present, although they are not likely to have a significant effect during blasting. These forces also oppose the external dynamic loads and are given by

$$-\mu \dot{\mathbf{u}}$$

where μ is a viscosity.

Both inertial and damping forces are volume distributed forces (i.e., forces per unit volume). Thus, using Equation 2.1, the body force vector, $\mathbf{b}$, is given by

$$\mathbf{b} = -\rho \mathbf{H} \ddot{\mathbf{u}} - \mu \mathbf{H} \dot{\mathbf{u}}$$

Substituting the above expression into Equation 2.8 and rearranging the result gives the equations of motion

$$\mathbf{M}\ddot{\mathbf{u}} + \mathbf{C}\dot{\mathbf{u}} = \mathbf{f} - \mathbf{q} \tag{2.9}$$

where

$$\mathbf{M} = \int_V \mathbf{H}^T \rho \mathbf{H} dV \tag{2.10}$$

$$\mathbf{C} = \int_V \mathbf{H}^T \mu \mathbf{H} dV \tag{2.11}$$

denote the mass and damping matrices, respectively.

Equation 2.9 has been derived for one element. Using the principle of superposition, the members of the element mass and damping matrices associated with each nodal point are added together (assembled) into global matrices. The vectors $\mathbf{f}$ and $\mathbf{q}$ are assembled in a similar manner.

In what follows, it will be assumed that the equations of motion refer to the assembled global system.

2.3 Mass and Damping Matrices, Internal Force Vector

Evaluation of M, C and q for an arbitrary quadrilateral element requires the use of numerical integration. This is easily accomplished by the use of isoparametric finite elements (Bathe, 1982).

Equations 2.2 and 2.3 may be used to transform an integral over an arbitrary quadrilateral in the (x, y) system to one over a standard rectangle $-1 \leq r, s \leq 1$ in the (r, s) coordinate system (see Figure 2.1). Gauss-Legendre quadrature may then be used to evaluate the integral.

The volume differential in Equations 2.6, 2.10 and 2.11 is given by $dV = |\mathbf{J}|drds$ where $|\mathbf{J}|$ denotes the determinant of the Jacobian J given in Equation 2.3. From Equation 2.10 the integral for $\mathbf{M}_{ij}$, a 2 by 2 submatrix of M, is therefore given by

$$\mathbf{M}_{ij} = \int_V \mathbf{H}_i^T \rho \mathbf{H}_j dV = \int_{-1}^{1}\int_{-1}^{1} \mathbf{H}_i^T \rho \mathbf{H}_j |\mathbf{J}| drds$$

Use of Gauss-Legendre quadrature to evaluate the above integral would result in a fully populated, but symmetric, matrix. For reasons that will become clear in Section 2.5, it is desirable to obtain a diagonal or lumped mass approximation to M. With the four point quadrilateral, this may be accomplished by means of a two point Gauss-Lobatto quadrature rule applied in both the r and s directions. Since, in Gauss-Lobatto integration, the integration points are at the ends of the intervals ($r = -1,1$; $s = -1,1$), and since the interpolation function h_i has the property $h_i(r_i, s_i) = 1$, the matrix product $\mathbf{H}_i^T \rho \mathbf{H}_j = \rho \mathbf{I}_2$, where $\mathbf{I}_2$ is the 2 by 2 identity matrix. A 2 by 2 diagonal of the lumped mass matrix is therefore

$$\mathbf{M}_i = \rho \mathbf{I}_2 |\mathbf{J}_i| \qquad 1 \leq i \leq 4 \tag{2.12}$$

This form of the lumped mass matrix has been implemented in the wave propagation code.

A lumped damping matrix is derived in exactly the same manner. As stated before, viscous damping has little effect during blasting. However, implementation of a silent outer boundary, composed of viscous dashpots which absorb rather than reflect wave energy, results in a diagonal damping matrix. This is discussed in Section 2.7.

The internal force vector, q, defined by Equation 2.6 is integrated over the element in a similar manner. For the ith node of an element, $1 \leq i \leq 4$, the vector $\mathbf{q}_i$ is given by

$$\mathbf{q}_i = \begin{pmatrix} q_{xi} \\ q_{yi} \end{pmatrix} = \int_V \mathbf{B}_i^T \mathbf{s} dV = \int_{-1}^{1}\int_{-1}^{1} \mathbf{B}_i^T \mathbf{s} |\mathbf{J}| drds \tag{2.13}$$

where the stresses are evaluated from Equation 2.5.

2.4 Load Vector

In blast modeling, the load vector f will be composed only of forces due to surface loads (tractions) generated in the blasthole by the blast. These forces are given by

$$\mathbf{f} = \int_S \mathbf{H}^T \mathbf{t} dS$$

where S is the surface over which the forces are acting and t are the tractions acting on S. In two dimensions S becomes the line L. The vector of forces at the nodes a and b in Figure 2.2 is

$$\mathbf{f} = \begin{pmatrix} f_{xa} \\ f_{ya} \\ f_{xb} \\ f_{yb} \end{pmatrix} = \int_L \begin{pmatrix} h_a & 0 \\ 0 & h_a \\ h_b & 0 \\ 0 & h_b \end{pmatrix} \begin{pmatrix} t_x \\ t_y \end{pmatrix} dL \tag{2.14}$$

where $h_a = \frac{1}{2}(1 - s)$ and $h_b = \frac{1}{2}(1 + s)$, $-1 \le s \le 1$.

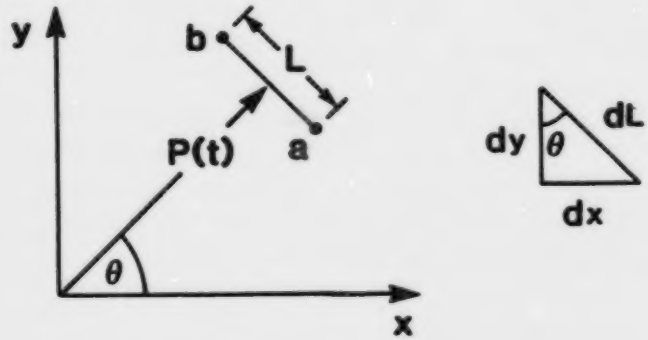


Figure 2.2
Surface loading composed of normal pulse load

The blast loading is usually given as a time dependent radial compressive stress $-p(t) = \sigma_{rr}(t)$; all other stresses are zero. Since the exterior unit normal to the blast hole is $\mathbf{n} = (n_r, n_\theta)^T = (-1, 0)$, the traction vector is

$$\mathbf{t} = \begin{pmatrix} t_r \\ t_\theta \end{pmatrix} = \begin{pmatrix} \sigma_{rr} & \sigma_{r\theta} \\ \sigma_{r\theta} & \sigma_{\theta\theta} \end{pmatrix} \begin{pmatrix} n_r \\ n_\theta \end{pmatrix} = \begin{pmatrix} p(t) \\ 0 \end{pmatrix}$$

where t_r and t_θ are the radial and shear components of the traction on the blasthole wall. Using the geometry of Figure 2.2, the traction vector in Cartesian coordinates is

$$\mathbf{t} = \begin{pmatrix} t_x \\ t_y \end{pmatrix} = \begin{pmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{pmatrix} \begin{pmatrix} t_r \\ t_\theta \end{pmatrix} = \begin{pmatrix} p(t)\cos\theta \\ p(t)\sin\theta \end{pmatrix}$$

From Figure 2.2, $dx = dL\sin\theta$ and $dy = dL\cos\theta$. Since $x = h_a x_a + h_b x_b$ and $y = h_a y_a + h_b y_b$,

$$\frac{dx}{ds} = \frac{x_b - x_a}{2} \qquad \frac{dy}{ds} = \frac{y_b - y_a}{2}$$

and the integral in Equation 2.14 may be transformed to the interval $-1 \le s \le 1$. The components of the vector $\mathbf{f}$ are therefore

$$\begin{pmatrix} f_{xa} \\ f_{ya} \\ f_{xb} \\ f_{yb} \end{pmatrix} = \frac{p(t)}{2} \begin{pmatrix} y_b - y_a \\ x_b - x_a \\ y_b - y_a \\ x_b - x_a \end{pmatrix} \tag{2.15}$$

From Equation 2.15 it may be seen that the load vector is a product of a time dependent function, $p(t)$, and a vector of geometrical functions. The geometrical functions may be computed once only during the mesh generation step and stored. The pressure pulse is evaluated during the solution stage. In this way multiple blastholes, each with a different blast pulse (e.g., a delay), may be easily handled.

2.5 Explicit Solution Scheme

Solution methods for Equation 2.9 are derived by substituting finite difference approximations for the derivatives $\ddot{\mathbf{u}}$ and $\dot{\mathbf{u}}$. Central finite difference approximations for the $\dot{\mathbf{u}}$ and $\ddot{\mathbf{u}}$ are

$$\dot{\mathbf{u}}_n = \frac{1}{2\Delta t}(\mathbf{u}_{n+1} - \mathbf{u}_{n-1})$$

$$\ddot{\mathbf{u}}_n = \frac{1}{(\Delta t)^2}(\mathbf{u}_{n+1} - 2\mathbf{u}_n + \mathbf{u}_{n-1})$$

where the subscript n denotes the time step ($t = n\Delta t$). These two equations may be combined to give

$$\mathbf{u}_{n+1} = \mathbf{u}_n + \Delta t\dot{\mathbf{u}}_n + \frac{(\Delta t)^2}{2}\ddot{\mathbf{u}}_n \tag{2.16}$$

$$\dot{\mathbf{u}}_{n+1} = \dot{\mathbf{u}}_n + \frac{\Delta t}{2}(\ddot{\mathbf{u}}_n + \ddot{\mathbf{u}}_{n+1}) = \mathbf{p}_{n+1} + \frac{\Delta t}{2}\ddot{\mathbf{u}}_{n+1} \tag{2.17}$$

The acceleration solution at time step $n+1$ is obtained by writing Equation 2.9 at time step $n+1$

$$\mathbf{M}\ddot{\mathbf{u}}_{n+1} + \mathbf{C}\dot{\mathbf{u}}_{n+1} = \mathbf{f}_{n+1} - \mathbf{q}_{n+1} \tag{2.18}$$

Substitution of Equations 2.16 and 2.17 into Equation 2.18 and rearranging the result gives the matrix equation

$$\left[\mathbf{M} + \frac{\Delta t}{2}\mathbf{C}\right]\ddot{\mathbf{u}}_{n+1} = \mathbf{f}_{n+1} - \mathbf{q}_{n+1} + \mathbf{C}\mathbf{p}_{n+1} \tag{2.19}$$

If the matrices M and C are both diagonal, solution of Equation 2.19 is trivial and does not require solution of a full system of equations. The method is therefore known as an explicit integration scheme. This particular implementation is known as a predictor-corrector scheme (Hughes, 1987, pp. 490-495) in that $\mathbf{p}_{n+1}$ is a predictor of $\ddot{\mathbf{u}}_{n+1}$ and Equation 2.17 is a corrector. The main advantage of this explicit scheme is that after execution of each step of the algorithm, field quantities are all evaluated at the same time step, $n+1$.

In a blasting problem, values of $\mathbf{u}_0$ and $\dot{\mathbf{u}}_0$ would be given (usually both zero). The algorithm requires a vector $\ddot{\mathbf{u}}_0$ which may be obtained from Equation 2.9 written at time step 0

$$\ddot{\mathbf{u}}_0 = \mathbf{M}^{-1}(\mathbf{f}_0 - \mathbf{C}\dot{\mathbf{u}}_0 - \mathbf{q}_0)$$

In practice, both the diagonal matrices M and C would be stored in separate vectors. The matrix product $\mathbf{C}\mathbf{p}_{n+1}$ is then easily performed. The vector q is computed on an element by element basis. The algorithm proceeds as shown in Figure 2.3.

The disadvantage of using an explicit scheme is that the method is conditionally stable. For stability the time step, Δt, must be smaller than a critical time step $\Delta t^e = T_0/\pi$, where T_0 is the smallest period that can be represented by the finite element mesh (Bathe, 1982, pp. 537-539).

T_0 is related to the smallest time required for elastic waves to traverse an element. Since $c_p > c_s$, Δt^e is given by

$$\Delta t^e = \frac{\alpha L}{c_p}$$

where α is some number less than unity and L is the smallest distance between adjacent nodes of an element. For the linear elements decribed in Section 2.3, $0.9 \leq \alpha \leq 1.0$ has been found to be suitable (Shantaram *et al*, 1976).

2.6 Constitutive Model for Fragmentation

Grady and Kipp (1980) developed a constitutive model for use in the analysis of fracture and fragmentation processes due to blast loads. A brief description of the model follows.

At any point in the material the components of the stress tensor σ_{ij} are related to the strain ϵ_{ij} by a modified Hooke's law

$$\sigma_{ij} = \lambda(1-D)\epsilon_{kk}\delta_{ij} + 2G(1-D)\epsilon_{ij}$$

The scalar D denotes the damage. Initially its value is zero and it can increase to a maximum of 1.0. The damage increases only when the hydrostatic stress is tensile, that is, $\sigma = 3B\epsilon > 0$. The growth of damage is described by the ordinary differential equation

Figure 2.3

Explicit Solution Algorithm - Linear Analysis

NT - number of time steps NE - number of elements

$$a_1 = \frac{\Delta t}{2} \qquad a_2 = \frac{(\Delta t)^2}{2}$$

Given $\mathbf{u}_0$ and $\dot{\mathbf{u}}_0$, compute $\ddot{\mathbf{u}}_0 = \mathbf{M}^{-1}(\mathbf{f}_0 - \mathbf{C}\dot{\mathbf{u}}_0 - \mathbf{K}\mathbf{u}_0)$
For $n = 0$ to NT-1 do
  time $= (n+1)\Delta t$
  $\mathbf{u}(= \mathbf{u}_{n+1}) \leftarrow \mathbf{u} + \Delta t\dot{\mathbf{u}} + a_2\ddot{\mathbf{u}}$
  $\dot{\mathbf{u}}(= \mathbf{p}_{n+1}) \leftarrow \dot{\mathbf{u}} + a_1\ddot{\mathbf{u}}$
  compute load vector $\mathbf{f}_{n+1}$
  $\mathbf{f}_{n+1} \leftarrow \mathbf{f}_{n+1} - \mathbf{C}\mathbf{p}_{n+1}$
  For $j = 1$ to NE do
    set $E = E_j$ and $\nu = \nu_j$
    get nodal displacements of element j, $\mathbf{u}_{n+1,j}$
    compute internal force vector $\mathbf{q}_{n+1,j}(E, \nu)$
    $\mathbf{f}_{n+1,j} \leftarrow \mathbf{f}_{n+1,j} - \mathbf{q}_{n+1,j}$
  End j
  $\ddot{\mathbf{u}}(= \ddot{\mathbf{u}}_{n+1}) \leftarrow (\mathbf{M} + a_1\mathbf{C})^{-1}\mathbf{f}_{n+1}$
  $\dot{\mathbf{u}}(= \dot{\mathbf{u}}_{n+1}) \leftarrow \dot{\mathbf{u}} + a_1\ddot{\mathbf{u}}$
End n

$\mathbf{f}_{n+1,j}$, $\mathbf{q}_{n+1,j}$ and $\mathbf{u}_{n+1,j}$ denote the components of $\mathbf{f}_{n+1}$, $\mathbf{q}_{n+1}$ and $\mathbf{u}_{n+1}$ corresponding to the element j.

$$\frac{dD}{dt} = \left[\frac{8\pi(m+3)k}{(m+1)(m+2)}\right]^{1/3} C_g \epsilon^{m/3} D^{2/3} \qquad \epsilon > 0 \tag{2.21}$$

where m and k are constants of the two parameter Weibull distribution, C_g is the crack propagation speed. These data can be determined from dynamic fracture tests on different rocks. Grady and Kipp (1980) used the values

$$m = 8$$
$$k = 1.7 \times 10^{27}/m^3$$
$$C_g = 1.3\text{km/s}$$

Thus, during the calculation of dynamic response due to a blast load, the material properties change if tensile stresses persist; otherwise they remain the same. This is the source of the nonlinear elastic behaviour in the model.

This model was implemented in the wave propagation code. It appeared to give reasonable results during the first few microseconds of a blast simulation. However, numerical instabilities (overflows) arose at later times. This happened in three geometrically different finite element models. It is not known whether the instabilities are related to the physical behaviour of the finite element model or to a problem in numerical precision.

Use of this model in the simulation of blasts in oil shales requires much care in the selection of the time step (Taylor *et al*, 1986). Taylor and Flanagan (1987) found that the time step must be varied as damage grows in the model and describe an automatic procedure for the selection of the time step as the simulation proceeds. The complexity of this procedure, and the fact that the model requires the use of parameters (m, k, and C_g) that are not easily measured, led to a decision to abandon this model in favour of a much simpler approach as described below.

The simpler model omits all elements from the internal force vector calculation if, at any time step, the tensile stress in the element exceeds the tensile strength of the rock. Once this occurs in an element, the element is considered to be permanently damaged and never again contributes to the internal force vector. As the simulation proceeds, elements surrounding a node will become 'deactivated', leaving the node without any internal resistance nor any means by which external forces can be transmitted to it. Motion of the node therefore stops and remains at the value computed at the time step when all surrounding elements failed. For this reason the time histories computed from this model are not representative. However, the damage pattern (the layout of the failed elements) is representative.

Implementation of the model is straightforward. After evaluation of the internal force vector, $\mathbf{q}$, of an element, the hydrostatic stress, σ, is computed at the centroid of the element. If this stress is tensile and exceeds the tensile strength of the element, the element is flagged and is never again used in the internal force vector calculation. At the end of the simulation, the plot of all failed elements is the damage pattern.

The main advantage of this approach is its simplicity and its use of a readily measured or estimated parameter, the tensile strength. It is also stable; the examples shown in Chapter 3 did

not exhibit any instabilities.

2.7 Silent Boundary

Radiation of energy from a blast propagates to large distances. However, in practice only a finite region of rock can be modelled. The result of this is that outwardly radiating energy is reflected from the outer boundaries of the model back into the medium. This introduces errors into the results.

One remedy to this problem is to cause the outer boundary to absorb energy incident from the medium rather than reflect it. Such boundaries are called transmitting, absorbing or silent. Lysmer and Kuhlemeyer (1969) proposed using viscous damping forces acting along the outer boundary to absorb energy. This type of silent boundary is incorporated in the wave propagation code.

The basic idea of the viscous boundary is illustrated in Figure 2.4. One applies boundary tractions t_{nb} and t_{sb} to an otherwise free boundary. These tractions are chosen to cancel those produced by the incoming waves, i.e.,

$$t_{ni} + t_{nb} = 0 \qquad t_{si} + t_{sb} = 0$$

where the subscript i denotes the incident fields. For the right and left boundaries shown in Figure 2.4, the boundary tractions are

$$t_{nb} = -\rho c_p \dot{u}_x \quad t_{sb} = -\rho c_s \dot{u}_y$$

while for the top and bottom boundaries

$$t_{nb} = -\rho c_p \dot{u}_y \quad t_{sb} = -\rho c_s \dot{u}_x$$

These are solutions to the one dimensional wave equation for the incident field (Achenbach, 1973). The signs are reversed so as to cancel the incident energy. The tractions are dissipative in nature and amount to placing a viscous dashpot at each node of the silent boundary.

Implementation of the silent boundary is done by adding the viscous damping forces to the force vector $\mathbf{f}$ of Equation 2.9. The methodology is discussed in Cohen and Jennings (1983) and is set out in detail below.

The force vector due to viscous damping at node a of Figure 2.5 is given by

$$\mathbf{f}_a = \begin{pmatrix} f_{xa} \\ f_{ya} \end{pmatrix} = \int_{y_a}^{y_b} \begin{pmatrix} h_a & 0 \\ 0 & h_a \end{pmatrix} \begin{pmatrix} t_{nb}(y) \\ t_{sb}(y) \end{pmatrix} dy$$

$$= \frac{-\rho \Delta y}{2} \int_{-1}^{1} \begin{pmatrix} h_a c_p & 0 \\ 0 & h_a c_s \end{pmatrix} \begin{pmatrix} h_a \dot{u}_{xa} + h_b \dot{u}_{xb} \\ h_a \dot{u}_{ya} + h_b \dot{u}_{yb} \end{pmatrix} ds$$

Using a two point Gauss-Lobatto integration rule results in

$$\mathbf{f}_a = \frac{-\rho \Delta y}{2} \begin{pmatrix} c_p & 0 \\ 0 & c_s \end{pmatrix} \begin{pmatrix} \dot{u}_{xa} \\ \dot{u}_{ya} \end{pmatrix} = -\mathbf{C}\dot{\mathbf{u}}_a$$

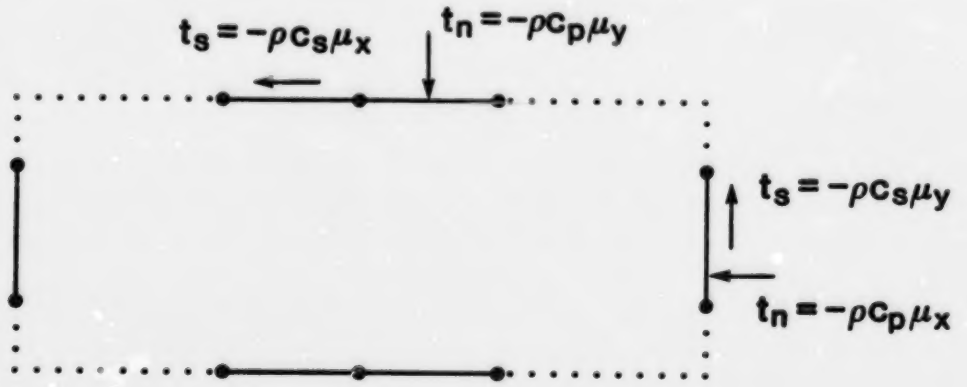


Figure 2.4

Silent boundaries of finite element mesh

where C is a lumped (diagonal) damping matrix. A similar formulation is made for node b. The above applies to nodes on the right or left boundaries. For node c on the top or bottom boundary (see Figure 2.5)

$$\mathbf{f}_c = \frac{-\rho \Delta x}{2} \begin{pmatrix} c_s & 0 \\ 0 & c_p \end{pmatrix} \begin{pmatrix} \dot{u}_{xc} \\ \dot{u}_{yc} \end{pmatrix} = -\mathbf{C}\dot{\mathbf{u}}_c$$

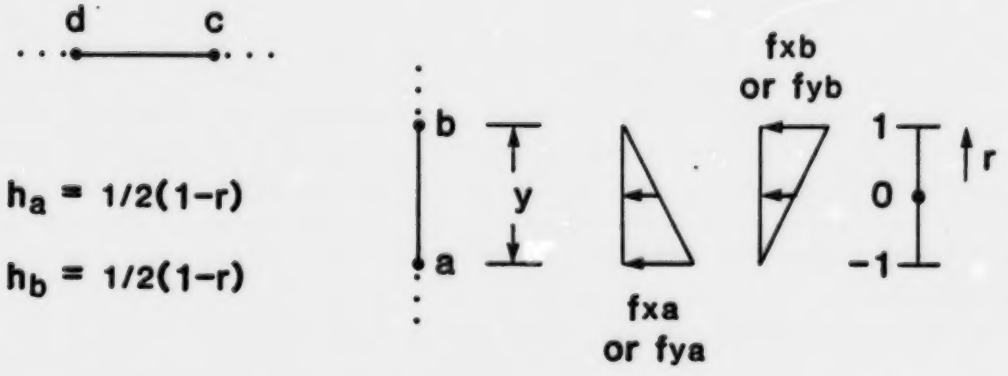


Figure 2.5

Applied forces for silent boundary

Thus addition of the damping forces to implement the silent boundary results in a nonzero damping term $C\dot{u}$ in Equation 2.9. The Gauss-Lobatto integration rule results in a lumped damping

matrix so that the simplicity of the explicit solution scheme discussed in Section 2.5 is preserved.

Cohen and Jennings (1983) found that the viscous silent boundary was both effective and inexpensive to use. In addition, it can be applied to problems where nonlinear material effects are present, although this should not be of concern since the outer boundaries of the model would likely be remote from nonlinear behaviour caused by a blast. Examples of the use of the silent boundary are given in Section 3.1.

2.8 Determination of the Equivalent Cavity

In Section 1.2 a qualitative model of rock blasting is discussed in which an annulus of cracked rock of radius r_b is formed around the blasthole. This annulus is known as the equivalent cavity. Computational blast modelling is naturally divided into near field models, in which the nonelastic behaviour within the equivalent cavity is modelled, and far field models, in which the rock is assumed to behave in a linear elastic manner. Near field models would be used to study comminution levels while far field models would be useful for determining ground motion to assess the potential for damage.

There exist significant differences between a near field and a far field model. The finite element mesh of a near field model would be quite fine in order to adequately model the details of the fragmentation process and would include the blasthole and the details of rock structure within the equivalent cavity. The mesh of a far field model would be coarser and would only include the region outside the equivalent cavity. The loading in a near field model would be the blasthole pressure time history whereas the loading of a far field model would be the traction time history at r_b due to the blasthole pressure time history.

For either of the above models it is necessary to determine the radius of the equivalent cavity. This may be done empirically using curves relating the 'radius of rupture' (i.e., radius of equivalent cavity) to blasthole diameter for different explosives (e.g., Calder, 1977). An alternative method is to use an empirical formula for r_b derived by Obert and Duvall (1950). In SI units, this formula is

$$r_b = 0.353W^{1/3} \tag{2.22}$$

where r_b is in meters and W is the weight of the explosives in kilograms.

The radius of the equivalent cavity would be used to determine the extent of a near field model. For a far field model, it is necessary to determine the traction time history at r_b due to the blasthole pressure. This is the boundary condition at $r = r_b$ in the far field model.

Determination of the traction on the wall of the equivalent cavity is not a simple matter. First of all, it is likely that the traction has both normal and shear components. Secondly, the form and magnitude of the traction are probably controlled by nonlinear material behaviour within the equivalent cavity. However, an upper limit for the normal traction on the cavity wall can be assumed to be the compressive strength of the rock. In addition, it is reasonable to assume that the rock within the equivalent cavity is pulverized to such an extent that the gases emanating from the explosion exert a pressure whose form is equivalent to that on the blasthole wall and do not

result in a significant shear traction. This is sufficient to determine the traction time history on the wall of the equivalent cavity. These ideas will be tested in Chapter 3.

CHAPTER 3

Applications

3.1 One Dimensional Test Problems

The first test problem is illustrated in Figure 3.1. A two dimensional halfspace ($x \geq 0$) is subjected to a uniform surface stress $p(t)$ given by

$$p(t) = \begin{cases} 0 & t \leq 0 \\ p_0 \sin(10\pi t) & t \geq 0 \end{cases}$$

where $p_0 = 200$ MPa. The analytical solution for velocity at x is (Achenbach, 1973, pp. 21-24)

$$\dot{u}(x,t) = \frac{c_p}{\lambda + 2G} p\left(t - \frac{x}{c_p}\right)$$

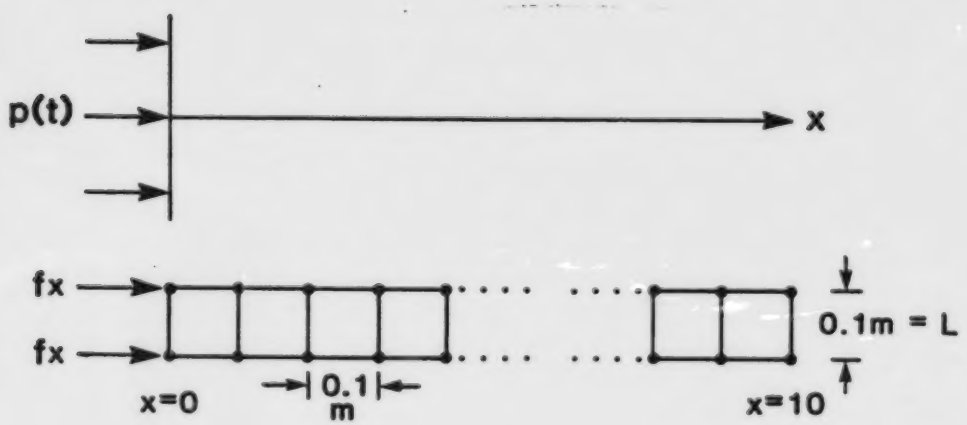


Figure 3.1

Finite element model - loaded halfspace

In this problem any plane parallel to the x axis is a plane of symmetry. Therefore the finite element model consists of a single row of elements whose nodes are restrained from movement in the y direction. One hundred square elements are used to model a 10 m portion of the halfspace (see Figure 3.1).

The tractions at the boundary $x = 0$ are given by

$$\begin{pmatrix} t_x \\ t_y \end{pmatrix} = \begin{pmatrix} -p(t) & 0 \\ 0 & 0 \end{pmatrix} \begin{pmatrix} -1 \\ 0 \end{pmatrix} = \begin{pmatrix} p(t) \\ 0 \end{pmatrix}$$

so that from Equation 2.15 the load vector for a node at $x = 0$ is

$$\begin{pmatrix} f_x \\ f_y \end{pmatrix} = \frac{L}{2} \begin{pmatrix} p(t) \\ 0 \end{pmatrix}$$

where $L = 0.1$ m is the width of an element.

The material properties and model parameters are given at the top of Table 3.1. The time step was chosen to be less than the time required for a P wave to travel the length of an element. The computed velocities at $x = 5$ m are also shown on Table 3.1 and compared with the analytical solution. Agreement is within 1%. Results for $t > 7.5$ ms include reflections from the boundary at $x = 10$ m.

To eliminate the reflections after $t > 7.5$ ms a silent boundary was placed at $x = 10$ m. That is, viscous forces were applied in the x direction at the two nodes on the right side of the mesh shown in Figure 3.1. The computed velocities at $x = 5$ m are shown on Table 3.2 and compared with the analytical solution. For this one dimensional problem, the silent boundary is an exact solution.

Table 3.1

Test Problem: Homogeneous Halfspace - Surface Load

$E = 6$ GPa $\quad \nu = 0.25 \quad \rho = 2000$ kg/m^3

$c_p \approx 1897.4$ m/s $\quad \Delta t = 5 \times 10^{-5}$ s

Velocity at $x = 5$ m

Time(ms)	Numerical	Analytical
2.5	0.0000	0.0000
3.0	0.6090	0.6040
3.5	1.4285	1.4317
4.0	2.2615	2.2590
4.5	3.0861	3.0859
5.0	3.9102	3.9119
5.5	4.7358	4.7370
6.0	5.5610	5.5609
6.5	6.3845	6.3834
7.0	7.2058	7.2044
7.5	8.0249	8.0236
8.0	8.9776	8.8409
8.5	10.6391	9.6559
9.0	12.2792	10.4686
9.5	13.9171	11.2786
10.0	15.5509	12.0859

Table 3.2

Test Problem: Homogeneous Halfspace - Surface Load

$E = 6$ GPa $\quad \nu = 0.25 \quad \rho = 2000$ kg/m^3

$c_p \approx 1897.4$ m/s $\quad \Delta t = 5 \times 10^{-5}$ s

Velocity at $x = 5$ m with silent boundary at $x = 10$ m

Time(ms)	Numerical	Analytical
2.5	0.0000	0.00000
3.0	0.6090	0.6040
3.5	1.4285	1.4317
4.0	2.2615	2.2590
4.5	3.0861	3.0859
5.0	3.9102	3.9119
5.5	4.7358	4.7370
6.0	5.5610	5.5609
6.5	6.3845	6.3834
7.0	7.2058	7.2044
7.5	8.0249	8.0236
8.0	8.8410	8.8409
8.5	9.6557	9.6559
9.0	10.4685	10.4686
9.5	11.2783	11.2786
10.0	12.0855	12.0859

3.2 Two Dimensional Examples

Blast Model 1

Figure 3.2 shows the finite element mesh of a one blasthole model called Blast Model 1. The borehole is centred at $x = y = 0$ and the lines $x = 0$ and $y = 0$ are lines of symmetry. The rock is homogeneous with the following material properties:

$$E = 3.1 \text{ GPa} \qquad \nu = 0.3 \qquad \rho = 2300 \text{ kg/m}^3 \qquad \sigma_t = 11 \text{ Mpa}$$

$$c_p \approx 1347 \text{ m/s}$$

where σ_t is the tensile strength. The time function of the pressure pulse on the blasthole wall is

$$p(t) = ae^{-\beta t}\sin(\omega t) \tag{3.1}$$

$$a = 1.327 \times 10^8 \text{ Pa}$$
$$\beta = 9.669 \times 10^3 \text{ s}^{-1}$$
$$\omega = 3.142 \times 10^3 \text{ s}^{-1}$$

This is plotted in Figure 3.3. The peak pressure is 15.6 Mpa at a time $t_m = 0.1$ ms. The purpose of this low peak pressure pulse is to study elastic wave propagation in the model and to assess the effect of the silent boundary.

Using a time step of $\Delta t = 5 \times 10^{-5}$ s for $NT = 320$ time steps, the displacements and velocities at a distance of approximately 0.5 m from the blasthole were computed. The top and right sides of the mesh were silent boundaries.

The results are shown on Figures 3.4 and 3.5. If the top and right sides of the mesh are not silent boundaries, the resulting velocities are shown on Figure 3.6. The arrival at about $t = 1.2 \times 10^{-2}$ s is a reflection from either the top or right boundary (or possibly both).

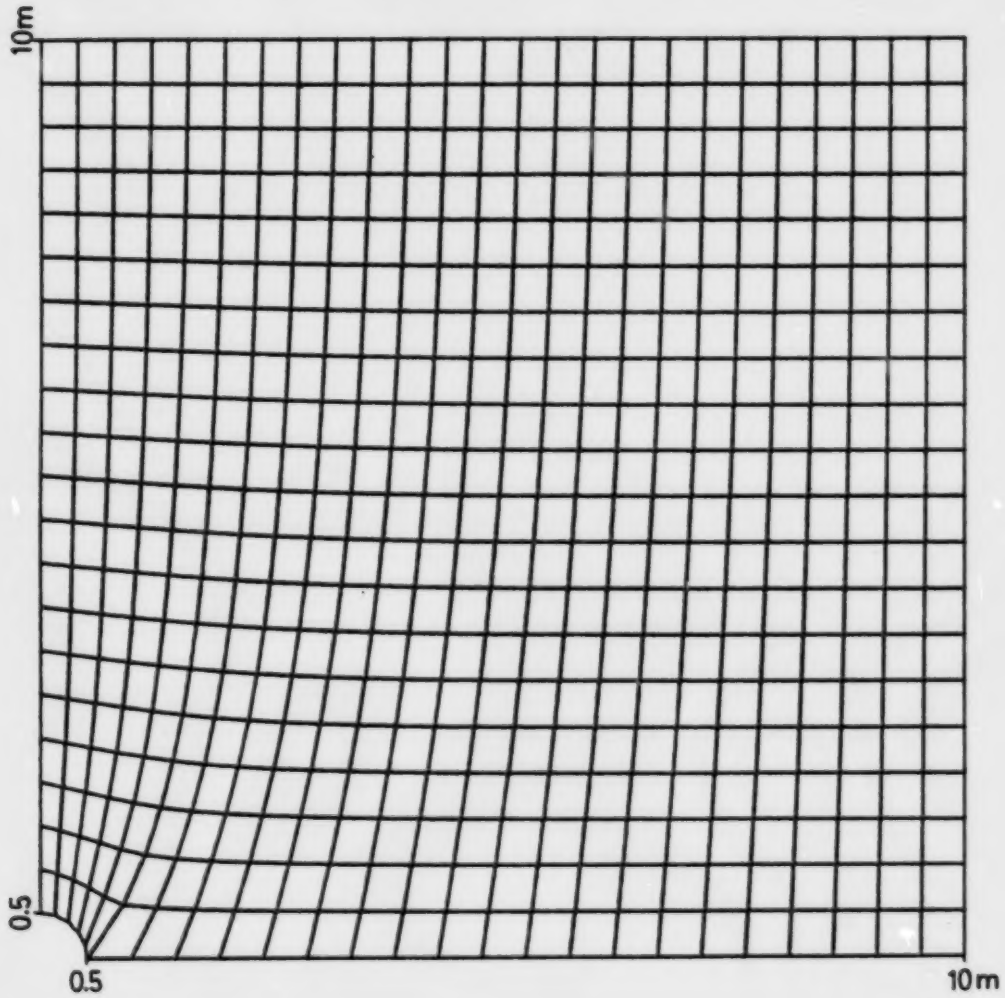


Figure 3.2
Blast Model 1 - finite element mesh

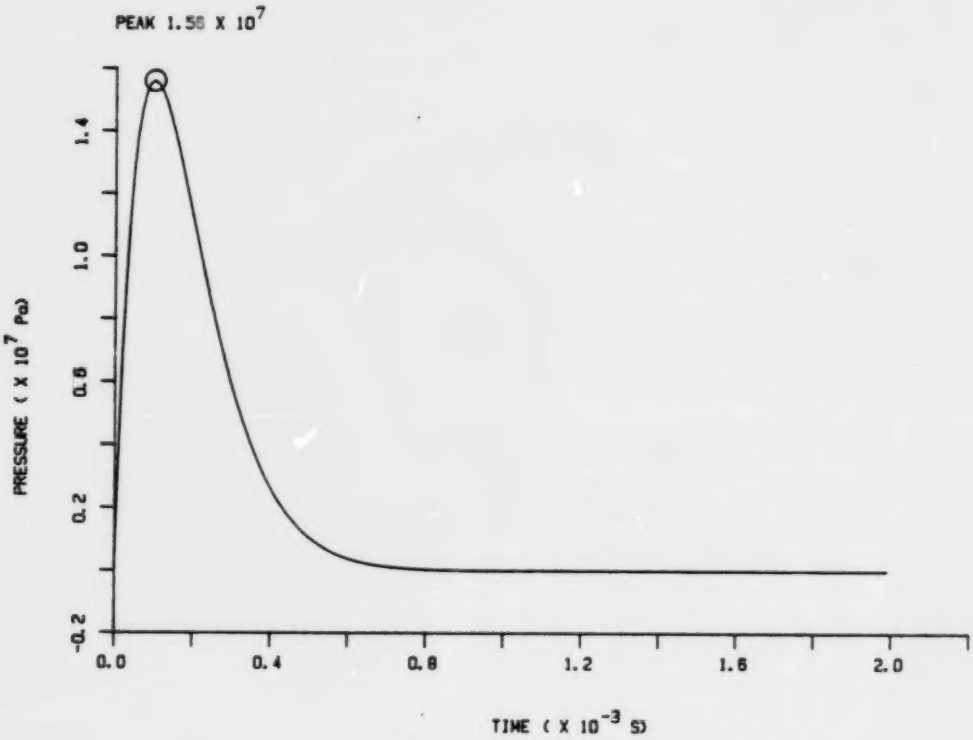


Figure 3.3

Blast Model 1 - pressure pulse on blasthole wall

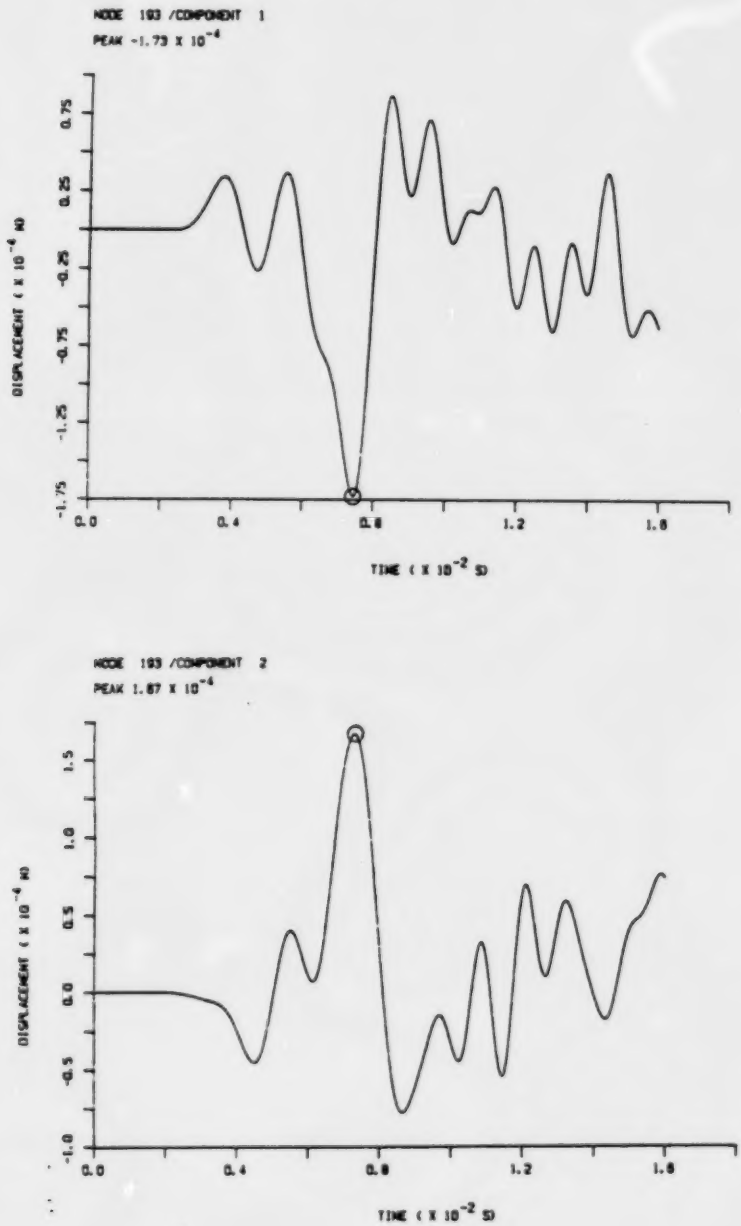


Figure 3.4
Blast Model 1 - displacements at 0.5 m distance

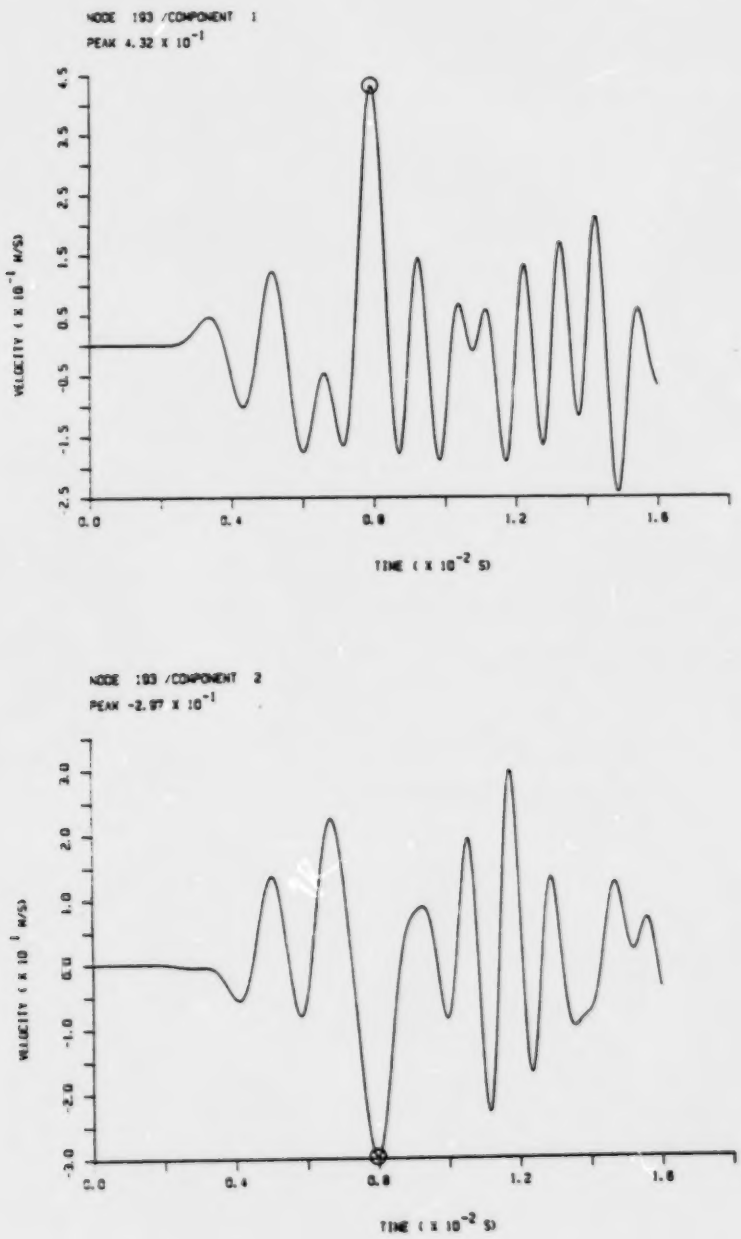


Figure 3.5
Blast Model 1 - velocities at 0.5 m distance

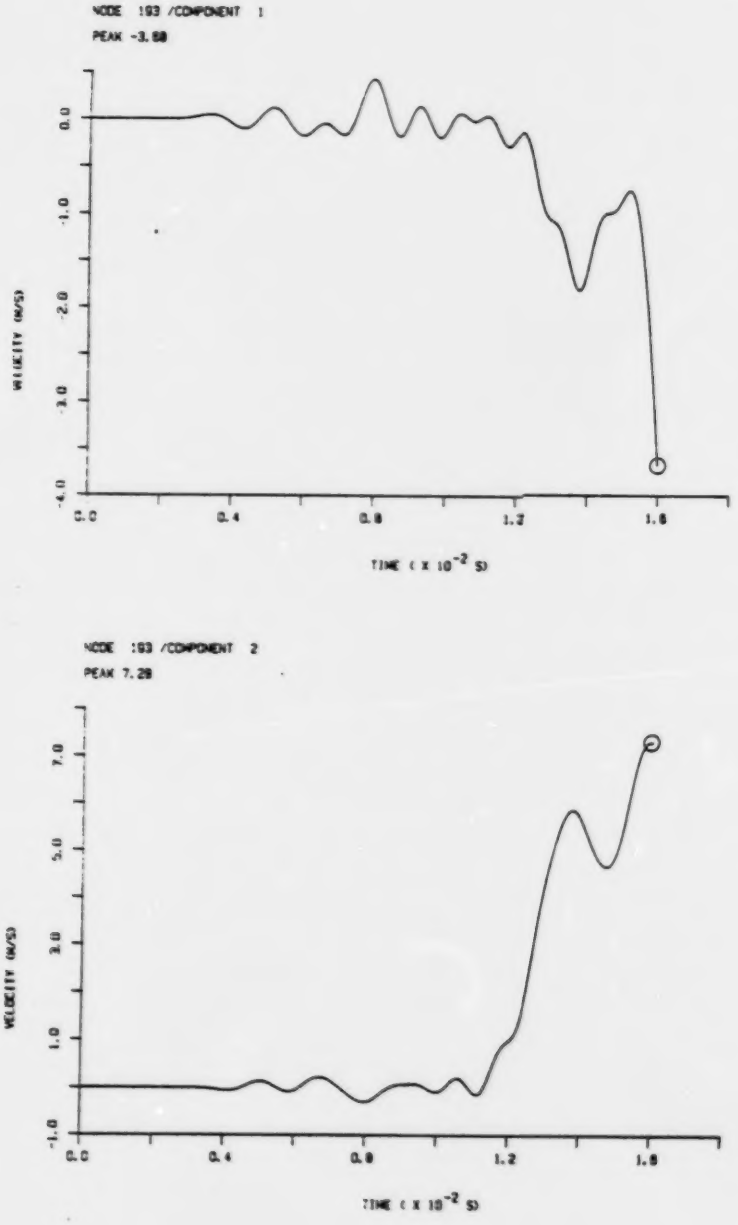


Figure 3.6

Blast Model 1 - velocities at 0.5 m distance. The top and right sides of the mesh are not silent boundaries.

Blast Model 2

The finite element mesh of Blast Model 2 is shown in Figure 3.7. It is essentially a rescaled version of Blast Model 1. The material properties are the same as those of Blast Model 1. The pressure pulse is of the same form as Equation 3.1, but with the parameters

$$\alpha = 2.995 \times 10^{10} \text{ Pa}$$
$$\beta = 9.669 \times 10^{3} \text{ s}^{-1}$$
$$\omega = 3.142 \times 10^{3} \text{ s}^{-1}$$

The peak pressure of this pulse is 3516 Mpa. This is the borehole pressure estimated from the detonation velocity of AQUAMEX (a CIL explosive) using the equation

$$P_m = N\rho V^2$$

where N is a constant, ρ is the specific gravity of the explosive, and V is the detonation velocity (Calder, 1977). For AQUAMEX, $\rho = 1.6$, $N = 1.4 \times 10^{-3}$ and $V = 4600 \times 3.28$ ft/s, giving $P_m = 509,931$ psi $= 3516$ Mpa.

The purpose of this model is to perform a damage analysis. A time step of $\Delta t = 5 \times 10^{-6}$ for $NT = 150$ time steps was used. The top and right sides were not silent boundaries. For a tensile strength of 11 Mpa, the damage pattern at $t = 0.75$ ms is shown in Figure 3.8. This pattern exhibits similarities to damage patterns calculated by Digby *et al* (1985) using the Grady and Kipp damage accumulation model. In particular, the diagonal slice of unfailed elements at left centre is also present in their models. However, it is possible that the pattern shown is an artefact of the geometry of the mesh used. This was not tested.

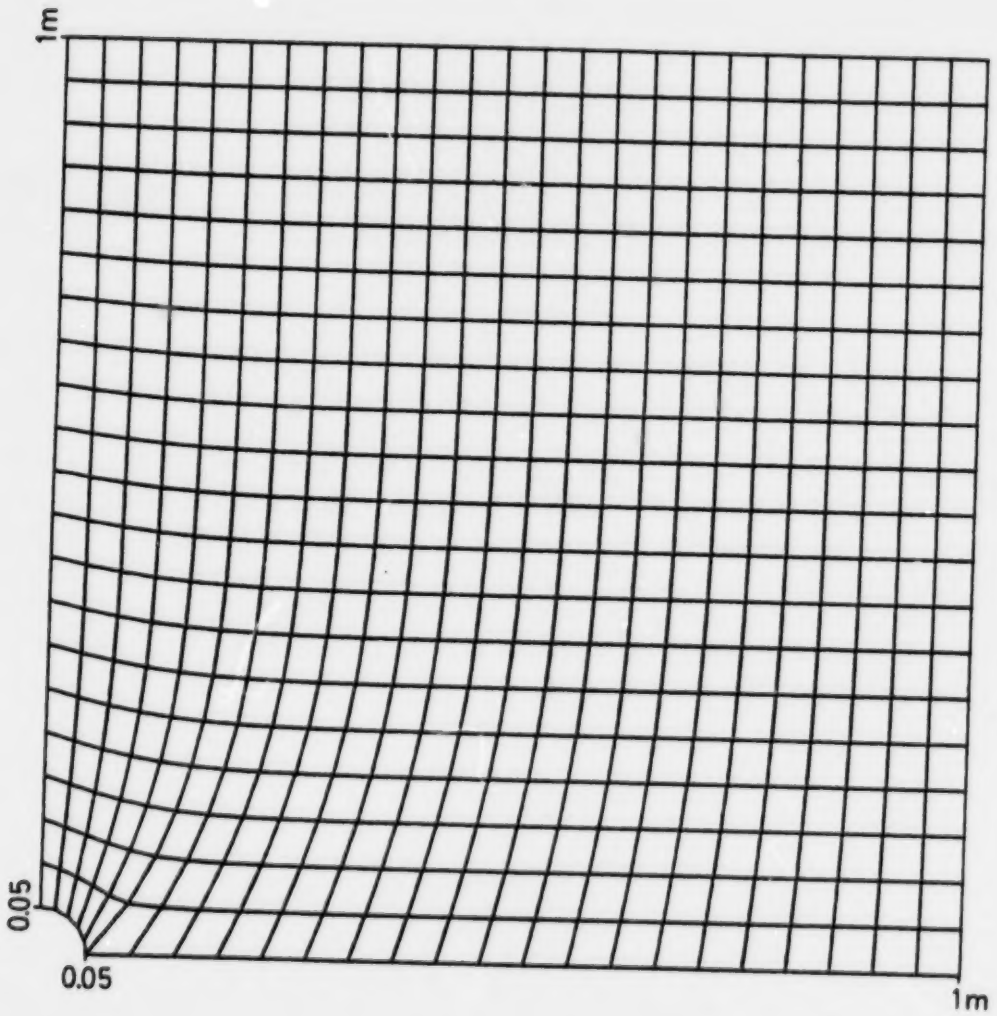


Figure 3.7
Blast Model 2 - finite element mesh

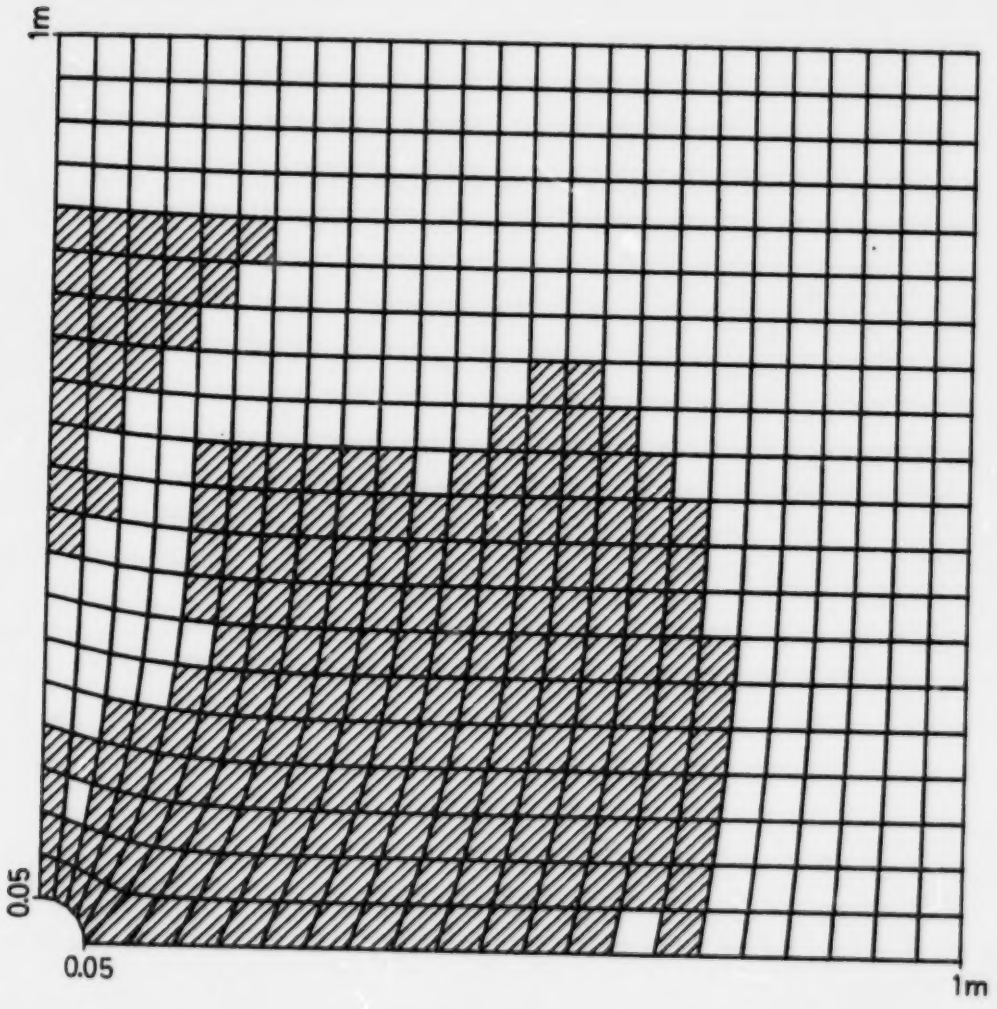


Figure 3.8
Blast Model 2 - damage pattern at $t = 0.75$ ms

3.3 Kidd Creek Mine Case Study

The Kidd Creek Mine is a massive sulphide orebody located 27 km north of Timmins, Ontario. During the 1970s several controlled blasts in the mine were monitored to obtain a criterion for blast-induced damage (Yu, 1980). As part of this work an empirical formula relating peak particle velocity to scaled range was developed. It is

$$V = 3.03\left(\frac{R}{W^{1/3}}\right)^{-1.33} \qquad \frac{R}{W^{1/3}} < 10 \tag{3.2}$$

where V is the peak particle velocity in m/s, R is the distance between the source and the observation point in m, and W is the charge size per delay in kg. W averaged about 1150 kg per 10 ms delay.

The blast geometries were typically very complex. The purpose of the modelling to be described below is to predict velocities due to a single blast and compare them with velocities computed from Equation 3.2. In order to perform this type of analysis, several assumptions must be made. They are:

1.) Wave propagation due to the blast may be modelled in two dimensions.
2.) The effects of geometrical complexities such as faults can be ignored.
3.) The blast itself can be modelled as a radial pressure pulse in a single vertical hole.

In addition, although the finite element code is entirely capable of modelling inhomogeneous media, the rock surrounding the blast is assumed to be homogeneous.

The host rock in the mine is a dense strong rhyolite. Faults in the mine produce contacts with an andesitic diorite. The properties of these two rock types are similar. The following values were assumed:

$$E = 80\text{GPa}$$
$$\nu = 0.2$$
$$\rho = 2900\text{kg/m}^3$$
$$\sigma_c = 135\text{MPa}$$

where σ_c is the uniaxial compressive strength. The compression wave velocity of this material is approximately 5536 m/s.

A far field model was used. The radius of the equivalent cavity, r_b, was estimated using Equation 2.22. For $W = 1150$ kg, $r_b \approx 4$ m. The pressure pulse was assumed to be the same form as that shown in Figure 3.3, but with a peak pressure equal to the uniaxial compressive strength of the rock. The parameters of this pulse are therefore

$$a = 1.149 \times 10^9 \text{ Pa}$$
$$\beta = 9.669 \times 10^3 \text{ s}^{-1}$$
$$\omega = 3.142 \times 10^3 \text{ s}^{-1}$$

The finite element mesh used is shown on Figure 3.9. Again, it is a scaled version of the mesh shown on Figure 3.2. Displacement, velocity and acceleration time histories were monitored at nodes 84, 112 and 140 of the mesh. The velocity time histories at each of these nodes are shown on Figures 3.10, 3.11 and 3.12. Comparison of the largest peak velocity at each of these nodes with the peak velocity predicted from Equation 3.2 is shown on Table 3.3. Considering the assumptions made, the comparison is favourable, especially at larger distances. No adjustments were made to the pressure pulse or the mesh geometry to obtain the results shown.

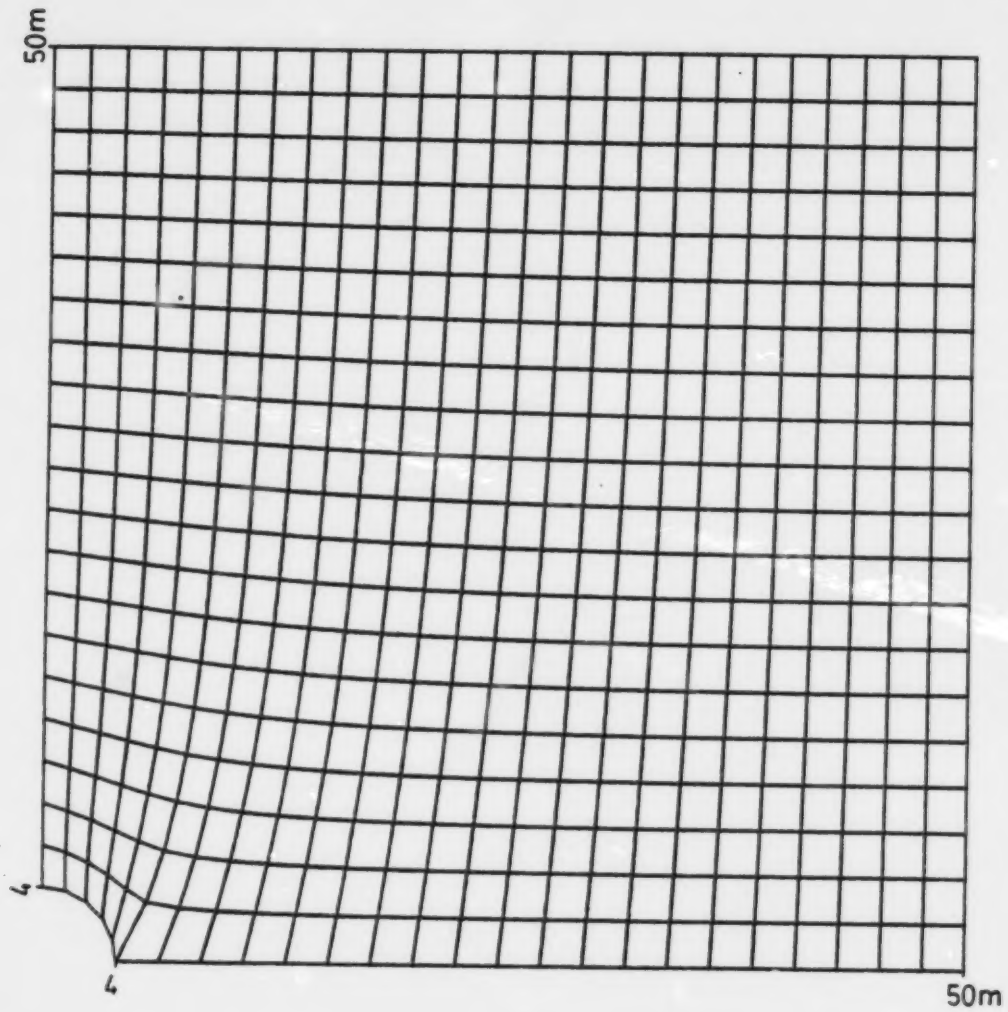


Figure 3.9
Kidd Creek Mine Case Study - finite element mesh

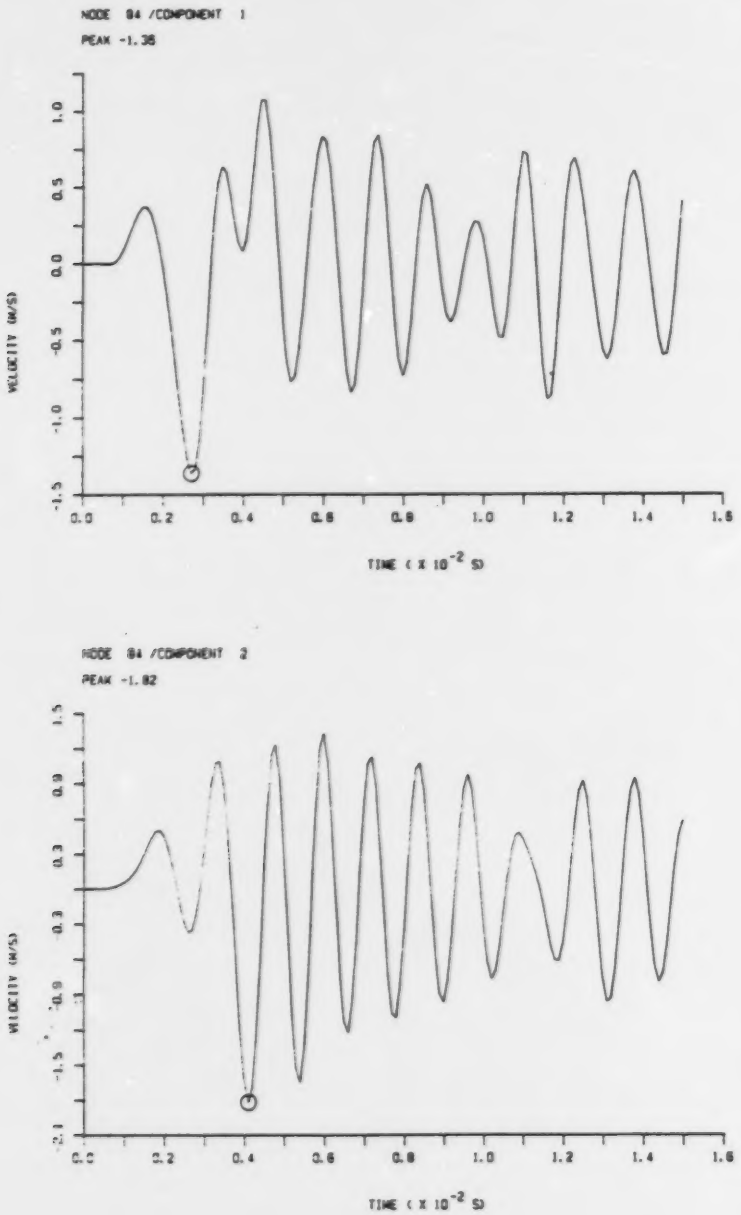


Figure 3.10
Kidd Creek Mine Case Study - velocities at node 84

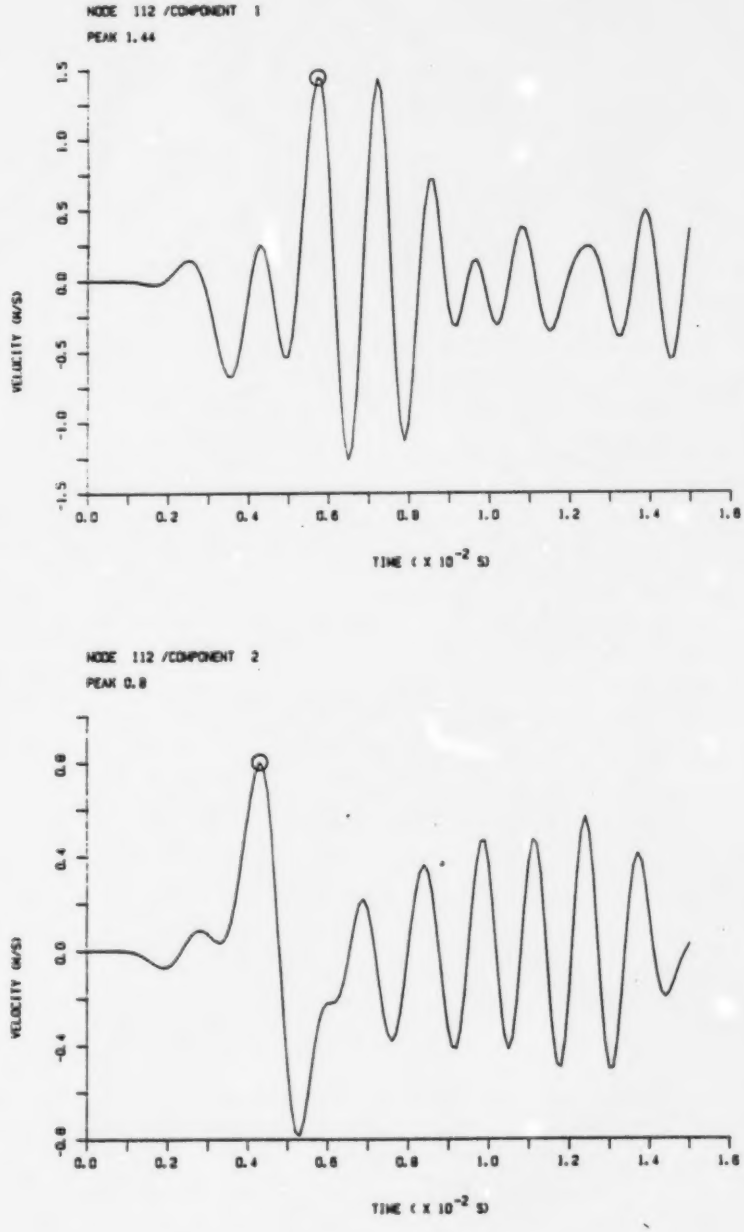


Figure 3.11
Kidd Creek Mine Case Study - velocities at node 112

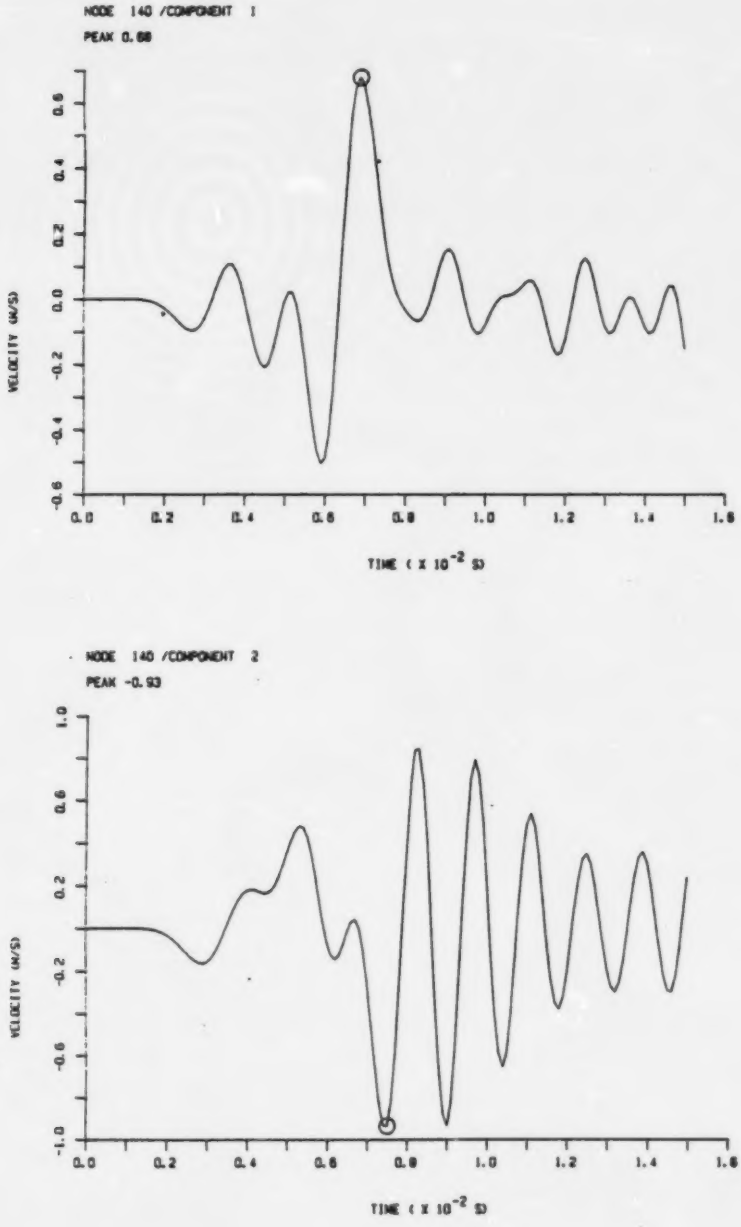


Figure 3.12
Kidd Creek Mine Case Study - velocities at node 140

Table 3.3

Kidd Creek Mine Case Study

Prediction of Peak Velocity

V = peak velocity computed by Equation 3.2 with W = 1150 kg

$\tilde{V}$ = largest peak velocity computed by program

Node	R(m)	V(m/s)	$\tilde{V}$(m/s)
84	11.35	2.61	1.82
112	15.66	1.42	1.44
140	20.38	0.87	0.93

CHAPTER 4

Conclusions and Recommendations

4.1 Conclusions

A two dimensional finite element wave propagation code, employing an explicit time integration scheme, has been written to model the effects of blasting in rock. A mesh generator and output display routines have also been supplied. A simple tensile failure model for fragmentation has been implemented in the code. An absorbing or silent boundary has been included to eliminate the effects of unwanted reflections from external boundaries of the finite element mesh. The finite element code has a restart facility. The entire package has been set up to operate on a microcomputer equipped with a math coprocessor and graphics capabilities.

The finite element code has been tested on some one and two dimensional problems and found to work satisfactorily. However, the analysis is time consuming. For example, on an IBM PS/2 Model 80, approximately half an hour is required for 300 time steps of a 500 node model.

Blast models are divided into near field and far field models. Near field models attempt to simulate the nonelastic behaviour which occurs near the blasthole and are useful for determining the level of comminution caused by a blast. Far field models ignore the material behaviour near the blasthole and are used to predict particle velocities or to study elastic wave propagation effects. Geometrically, the division between near and far field models is a radius, r_1, known as the radius of the equivalent cavity. At distances greater than r_1 the rock is assumed to behave elastically .

One of the more complicated aspects of blast modelling is determining a reasonable method of representing the source. For a near field model a compressive pulse with a large short duration peak acting on the blasthole wall was used. For a far field model a similar shaped pulse acting on the wall of the equivalent cavity, but with a peak equal to the compressive strength of the rock can be used as the source. In addition to the form of the source, the radius of the equivalent cavity must be estimated for a far field model. This was done using empirically derived equations.

The above ideas for far field models were used in an attempt to predict the peak velocities due to a blast and compare them with an equation empirically derived from measurements made during controlled blasts in the Kidd Creek Mine. The predictions were in reasonable agreement, despite the number of assumptions made. However, there is still a considerable amount of uncertainty in this approach to estimating the source characteristics of a far field model.

4.2 Recommendations

The following projects are recommended:

- Better methodologies for estimating the source characteristics of both near and far field models should be developed. More empirical observations would be necessary to develop such techniques.
- Further research should be done into the simple tensile failure model. Although the model is appealing because of its simplicity, the effects of mesh size and geometry are not understood.
- More case studies should be used to test the model and code. The more the code is used, the more will be learned of its utility and limitations.
- An implicit time integration scheme should be implemented, at least for linear far field models. The reason for this is that the use of an explicit time integration scheme requires a small time step and therefore, on a microcomputer, long execution times. Since most problems do not require extremely large amounts of storage (especially if the absorbing boundary is used), it is possible to contemplate an implicit scheme which would not require a small time step.
- Development of a three dimensional model could be considered. This may require a larger machine than a microcomputer or, alternatively, a microcomputer with an operating system that allows access to all available storage.

No particular priority is given to any of the above. The fourth project (development of an implicit scheme) would require the least amount of time. If any priority is to be assigned, it should be to the third item, the case studies.

REFERENCES

Achenbach, J. D., 1973. *Wave Propagation in Elastic Solids*, North Holland.

Bathe, K. J., 1982. *Finite Element Procedures in Engineering Analysis*, Prentice-Hall.

Bhandari, S., 1975. *Studies on Rock Fragmentation in Blasting*, Ph. D. Thesis, University of New South Wales.

Blair, D. P., 1981. Finite element modelling of ground surface displacements due to underground blasting, *International Journal for Numerical and Analytical Methods in Geomechanics*, 5, 97-113.

Bolt, B. A. and Smith, W. D., 1976. Finite element computation of seismic anomalies for bodies of arbitrary shape, *Geophysics*, 41, 145-150.

Brady, B. H. G. and Brown, E. T., 1985. *Rock Mechanics for Underground Mining*, George Allen & Unwin.

Buell, W. R. and Bush, B. A., 1973. Mesh generation - a survey, *Journal of Engineering for Industry, ASME*, 95, 332-338.

Burleigh, D. W., 1987. Three FORTRAN77 Compilers, *BYTE*, 12(13), 187-194.

Calder, P., 1977. *Pit Slope Manual Chapter 7 - Perimeter Blasting*, Canada Centre for Mineral and Energy Technology, Report No 77-14.

Carey, G. F. and Oden, J. T., 1984. *Finite Elements - Computational Aspects, Volume III*, Prentice-Hall.

Cohen, M. and Jennings, P. C., 1983. Silent boundary methods for transient analysis, in *Computational Methods in Mechanics, Volume 1 - Computational Methods for Transient Analysis*, (T. Belytschko and T. J. R. Hughes, eds.), Elsevier.

Constantino, C. J., 1967. Finite element approach to stress wave problems, *Journal of the Engineering Mechanics Division, ASCE*, 93, 153-176.

Constantino, C. J., Wachowski, A. and Barnwell, U. L., 1967. *Finite element solution for layered media caused by nuclear detonation*, Proceedings International Symposium on Wave Propagation and Dynamic Properties of Earth Materials, University of New Mexico.

Cunningham, C., 1983. *The Kuz-Ram model for prediction of fragmentation from blasting*, Proceedings 1st International Symposium on Rock Fragmentation by Blasting, Lulea, Sweden.

Dienes, J. K. and Margolin, L. G., 1980. A computational approach to rock fragmentation, in *Proceedings 21st U. S. Symposium on Rock Mechanics*, (D. A. Summers, ed.), University of Missouri, Rolla.

Digby, P. J., Nilsson, L. and Oldenburg, M., 1985. Finite element simulation of time dependent fracture and fragmentation processes in rock blasting, *International Journal of Rock Mechanics and Mining Science*, **9**, 317-329.

Duvall, W. I., 1953. Strain wave shapes in rock near explosions, *Geophysics*, **18**, 310-323.

Grady, D. E. and Kipp, M. E., 1980. Continuum modelling of explosive fracture in oil shale, *International Journal of Rock Mechanics and Mining Science*, **17**, 147-157.

Harries, G., 1977. Theory of blasting, in *Drilling and Blasting Technology*, (Australian Mineral Foundation), Adelaide.

Heller, L. W., 1972. *Finite element analysis of the response of Rifle Gap dam to the effects of Rulison underground detonation*, Proceedings Symposium U.S. Army Engineers Waterways Experiment Station.Vicksburg

Herrmann, L. R., 1976. Laplacian-isoparametric grid generation scheme, *Journal of the Engineering Mechanics Division, ASCE*, **102**, 749-756.

Heusinkveld, M., Bryan, J., Burton, D., and Snell, C., 1975. *Controlled blasting calculations with the TENSOR-74 code*, Publication No. UCRL-51740, Lawrence Livermore Laboratory, Berkeley, California.

Hughes, T. J. R., 1987. *The Finite Element Method*, Prentice-Hall.

INCO Ltd., 1987. *Research and Development in Manitoba Mines, Task 5 - Optimization of Blast Design for Blasthole Mining in Manitoba Mines, Literature Survey/Final Report*, Canada Centre for Mineral and Energy Technology, Project No. 5-9251.

Ingraffea, A. R., 1977. *Discrete Fracture Propagation in Rock*, Ph. D. Thesis, University of Colorado, Boulder.

Kutter, H. K., 1967. *The Interaction Between Stress Wave and Gas Pressure in an Underground Explosion in Rock*, Ph. D. Thesis, University of Minnesota.

Kuznetsov, V. M., 1973. The mean diameter of the fragments formed by blasting rock, *Soviet Mining Science*, **9**, 144-148.

Langfars U. and Kihlstrom, B., 1978. *Rock Blasting*, John Wiley & Sons.

Lysmer, J. and Kuhlemeyer, R. L., 1969. Finite dynamic model for infinite media, *Journal of the Engineering Mechanics Division, ASCE*, **95**, 859-877.

Margolin, L. G., 1981. *Calculations of cratering experiments with the bedded rock model*, Proceedings American Physical Society Symposium on Shock Waves in Condensed Matter, Menlo Park, California.

Nilsson, L. and Oldenburg, M., 1983. Nonlinear wave propagation in plastic fracturing materials - a constitutive modelling and finite element analysis, in *Proceedings IUTAM Symposium, Tallin (Nonlinear Deformation Waves)*, (U. Nigul and J. Engelbrecht, eds.), Springer-Verlag.

Obert, L. and Duvall, W. I., 1950. *Generation and Propagation of Strain Waves in Rock - Part 1*, U. S. Bureau of Mines Report of Investigations 4683.

Porter, D. D., 1970. *A Role of Borehole Pressure in Blasting: The Formation of Cracks*, Ph. D. Thesis, University of Minnesota.

Rosin, R. and Rammler, E., 1933. Laws governing fineness of powdered coal, *Journal of the Institute of Coal*, **7**, 29-36.

Santich, J. R., 1976. *Dynamic Models of Rock Blasting*, Ph. D. Thesis, University of New South Wales.

Shantaram, D., Owen, D. R. J. and Zienkiewicz, O. C., 1976. Dynamic transient behaviour of two and three dimensional structures including plasticity, large deformation effects and fluid interaction, *Earthquake Engineering and Structural Dynamics*, **4**, 561-578.

Sharpe, J. A. , 1942. The production of elastic waves by explosion pressures - I. Theory and empirical field observations, *Geophysics*, **3**, 144-165.

Shastri, S. M. and Malumdar, S., 1982. A dynamic finite element analysis of stress wave propagation and rock fragmentation in blasting, in *Numerical Methods in Geomechanics*,(Z. Eisenstein, ed.), A. A. Balkema.

Shipley, S. A., Leistner, H. G. and Jones, R. F., 1967. *Elastic wave propagation - a comparison between finite element predictions and exact solutions*, Proceedings International Symposium on Wave Propagation and Dynamic Properties of Earth Materials, University of New Mexico.

Smith, W. D., 1975. The application of finite element analysis to body wave propagation problems, *Geophysical Journal of the Royal Astronomical Society*, **42**, 747-768.

Starfield, A. M. and Pugliese, J. M., 1968. Compression waves generated in rock by cylindrical explosive charges, *International Journal of Rock Mechanics and Mining Science*, **5**, 65-77.

Sweagle, J. W., 1978. *1978*, TOODY-IV, A computer program for two dimensional wave propagation.Publication No. SAND-18-0552, Sandia Laboratories, Albuquerque, New Mexico

Taylor, L., Chen, E.P. and Kuszmaul, J. S., 1986. Microcrack-induced damage accumulation in brittle rock under dynamic loading, *Computer Methods in Applied Mechanics and Engineering*, **55**, 301-320.

Taylor, L. M. and Flanagan, D. P., 1987. *PRONTO-2D, A Two Dimensional Transient Solid Dynamics Program*, Sandia National Laboratories, Report No. SAND86-0594.

Verner, E. A., 1971. *Finite Element Stress Formulation for Wave Propagation*, Ph. D. Thesis, University of Texas, Austin.

White, W., Valliappan, S. and Lee, I. K., 1977. Unified boundary for finite dynamic models, *Journal of the Engineering Mechanics Division, ASCE*, **103**, 949-964.

Yu, T. R., 1980. Ground Control at Kidd Creek Mine, in *Underground Rock Engineering*, (Canadian National Committee on Rock Mechanics, ed.), CIM Special Volume 22.

APPENDIX A

Elastic Constitutive Model

Elastic Constants

E - Young's modulus
ν - Poisson's ratio
ρ - density
B - bulk modulus
λ - Lame's constant
G - shear modulus

$$\lambda = B - \frac{2}{3}G = \frac{E\nu}{(1+\nu)(1-2\nu)}$$

$$G = \frac{E}{2(1+\nu)} \qquad B = \frac{E}{3(1-2\nu)}$$

$$c_p = \sqrt{\frac{E(1-\nu)}{(1+\nu)(1-2\nu)\rho}} \qquad c_s = \sqrt{\frac{E}{2(1+\nu)\rho}}$$

Hooke's law

Compressive stresses are assumed negative; tensile stresses positive.

σ_{ij} - stress $\qquad$ ϵ_{ij} - strain

$$\sigma_{ij} = \lambda\epsilon_{kk}\delta_{ij} + 2G\epsilon_{ij}$$
$$= B\epsilon_{kk}\delta_{ij} + 2G(\epsilon_{ij} - \tfrac{1}{3}\epsilon_{kk}\delta_{ij})$$

$$\epsilon_{kk} = \begin{cases} \epsilon_{xx} + \epsilon_{yy} + \epsilon_{zz} & \text{in three dimensions} \\ \epsilon_{xx} + \epsilon_{yy} & \text{in two dimensions} \end{cases}$$

$$\sigma_{kk} = 3(\lambda + \frac{2}{3}G)\epsilon_{kk} = 3B\epsilon_{kk}$$

APPENDIX B

User's Manual

B.1 Introduction

The program package consists of three modules, a pre-processor, a finite element processor and a post-processor, each composed of one or more programs. Each of the programs operates independently of the others. No attempt is made to integrate the programs into one large package since the manner in which this would be done is problem dependent. All the programs are written in standard FORTRAN77.

All program development was done using an IBM PS/2 Model 80 microcomputer equipped with a math coprocessor. The Lahey FORTRAN77 compiler was used to develop the programs. Although compilation time with this compiler is very fast (hence its suitability for program development), execution time tends to be slower than say Microsoft FORTRAN or Ryan-McFarland FORTRAN (Burleigh, 1987). However, since there are no nonstandard features in any of the modules, each should compile without error using another compiler. The finite element processor was tested on the Amdahl 470 at the University of British Columbia. No problems with compilation or execution were encountered.

The graphics programs use the Graphics Development Toolkit graphics routines distributed by Graphics Software Systems of Beaverton, Oregon. These routines consist of drivers for various 'workstations' such as a screen or plotter, and a language 'binding' for the FORTRAN compiler being used.

The programs are:

Pre Processors

MGEN - mesh and data generator for finite element program
DMESH - displays finite element mesh

Finite Element Processor

BLAST - main driver for finite element program
BSUB - subroutines used in finite element analysis

Post Processors

PLOT - plots time histories computed by BLAST-BSUB
DDAM - displays damage pattern computed by BLAST-BSUB

The operation of each of these programs is described in the following sections. All of the programs are interactive, requiring a minimum of data input.

B.2 Program MGEN

MGEN generates the finite element mesh for a one blasthole, one material model of the type shown in Figure B.1. Although the capabilities of MGEN are rather limited, it could be easily modified to generate more complicated models. The program is stored in the file MGEN.FOR and must be compiled and linked to form an executable module, MGEN.EXE (or some other name, if desired). The program is entirely interactive. Figure B.2 shows a typical session which generates the mesh shown in Figure 3.2 and corresponding data.

Note that the number of nodes in the x direction must be greater than the number of nodes in the y direction. This is because the number of nodes describing the blasthole, NB, is computed as

$$NB = NX - NY + 1$$

The minimum and maximum x and y coordinates, Young's modulus, Poisson's ratio, density and borehole radius must all be given in a consistent set of units.

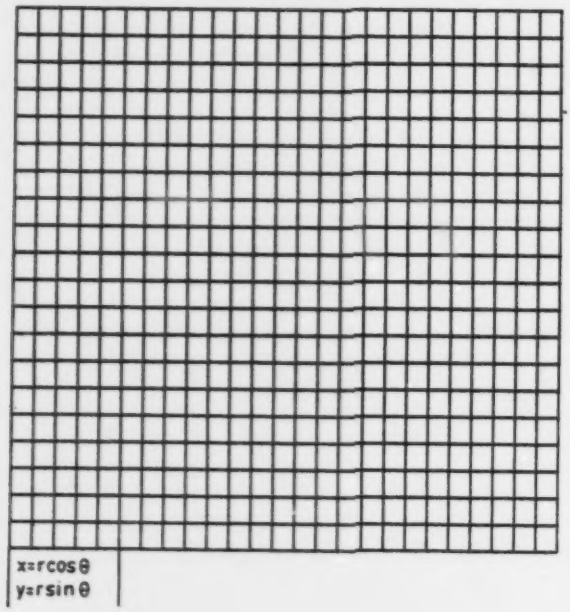
In MGEN the mesh is initially assumed to be regular in both the x and y directions and is generated in that manner. Nodal coordinates of the mesh are generated by setting the nodal coordinates on the periphery Γ of the mesh to their desired values. The two boundary value problems

$$\begin{aligned} \nabla^2 x &= 0 \\ x &= x_b \text{ on } \Gamma \\ \nabla^2 y &= 0 \\ y &= y_b \text{ on } \Gamma \end{aligned}$$

are then solved i.e., the nodal coordinates are the solution to a steady state flow problem with Dirichlet boundary conditions. Details of the generation of such Laplacian grids can be found in Carey and Oden (1984), Herrmann (1976) and Buell and Bush (1973). The above two problems are solved by iterative techniques using finite differences. The solution to Laplace's equation is computed using an iterative finite difference solver. This requires that a solution tolerance be provided. Usually 10^{-3} is sufficient.

A more elaborate mesh generator could be devised. However, without the use of interactive graphics (e.g., a mouse), such a program would be cumbersome.

Output from MGEN consists of a binary file MESH.DAT to be read by the finite element processor and an ASCII file MGEN.OUT which lists all the input data and the coordinates of the finite element mesh. The contents of MESH.DAT are described in Section B.4. The radial distance of each coordinate from the origin as well as the boundary condition codes (see Section B.4) are printed in MGEN.OUT.



$$\nabla^2 x = 0 \qquad \nabla^2 y = 0$$

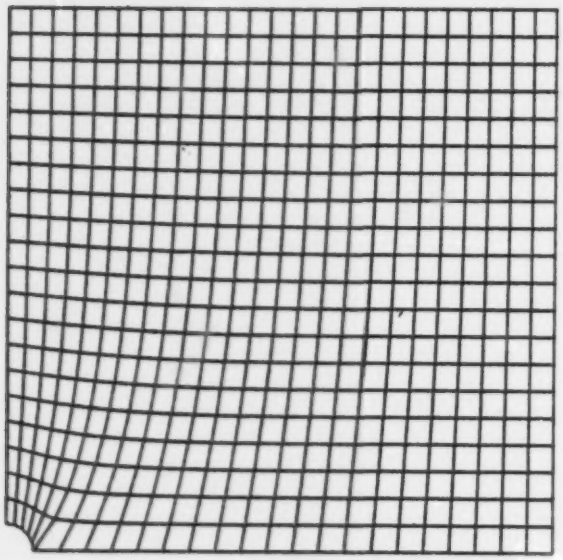


Figure B.1
Generation of mesh by MGEN

```
C:\BLAST>MGEN

Job title in quotes (maximum 80 characters):
'BLAST MODEL 1'

Number of nodes in x direction: 26

Number of nodes in y direction: 21

Minimum and maximum x coordinate: 0.0 1.0

Minimum and maximum y coordinate: 0.0 1.0

Youngs modulus 22.5E9

Poissons ratio 0.25

Density 2640

Tensile strength 11.E6

Borehole radius: 0.05

Tolerance for grid coordinates: 1.E-3

Number of iterations for x coordinates:          23
Number of iterations for y coordinates:           9

C:\BLAST>
```

Figure B.2
Sample MGEN session

B.3 DMESH

DMESH displays the finite element mesh using the coordinates and material properties in the file MESH.DAT. The mesh may be displayed on a screen, a printer or a plotter. Up to six material types may be displayed, each denoted by a hatch pattern.

The program is stored in a file called DMESH.FOR and is made into an executable module by first compiling it and then linking it to the library of graphics routines associated with the particular FORTRAN compiler in use. For the Lahey compiler, the LINK statement which forms the executable module is

LINK DMESH, , , LAHEYCGI

which results in an executable module called DMESH.EXE. The correct drivers for the screen, printing and plotting devices must be loaded before executing DMESH.

Whatever the display device used, the user is prompted for a scale factor, SF, in the range $0 \leq SF \leq 1$ which determines the portion of the display area which will be used to display the mesh. Scale factors less than 1.0 are often useful on hard copy devices. The mesh is first displayed on the screen. The following commands may then be issued:

R (or r) - display mesh on printer
L (or l) - display mesh on plotter
Q (or q) - stop and return to DOS

B.4 Finite Element Processor

These routines perform the finite element analysis of a wave propagation problem as described in Chapter 2. The program reads all the required data from a binary file, MESH.DAT, (created by MGEN or another routine) and sets up the arrays and files required for the analysis.

The program assumes that the nodes are located on a grid composed of NX nodes in the x direction and NY nodes in the y direction. The grid may be irregularly shaped. This approach avoids the need to supply another integer array of element connections - what nodes make up each element. These connections are necessary when computing element matrices. If the elements and nodes are numbered as shown in Figure B.3, then the element number is given by

$$N = I + (J - 1) \times (NX - 1)$$

Given N and NX, the I and J indices of the left lower node of each element may be computed. From this the nodes making up each element may be determined as necessary.

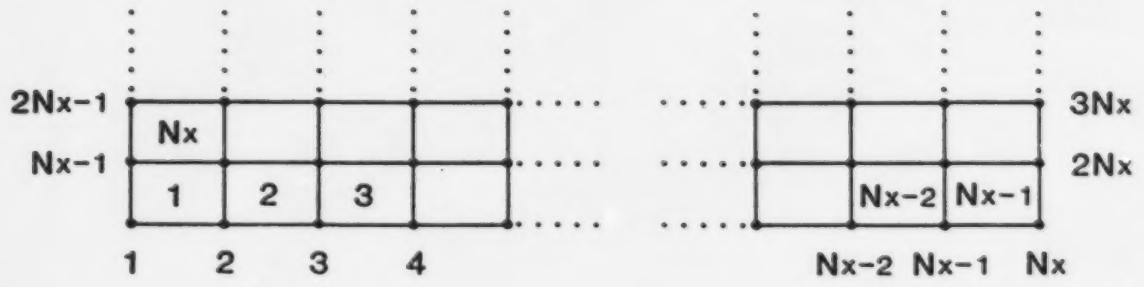


Figure B.3

Element and node numbering system

The contents of MESH.DAT are shown on the next page. Other required data are obtained in an interactive manner. Either simple numbers or yes/no answers must be supplied. The answer yes is given by the letter Y(y) and the answer no is given by the letter N(n). Both letters can be given in either upper or lower case; After reading MESH.DAT, the job title is printed on the screen and the user is then asked

Is this a restart?

If the answer is no, the user is then prompted as described below.

Type number of time steps

Type time step

Are there any silent boundaries in the model?

Contents of File MESH.DAT

RECORD 1 - TITLE

80 columns of job title or description.

RECORD 2 - NB,NE,NN,NX,NY,NM

NB - number of nodes where tractions are prescribed
NE - number of elements
NN - number of nodes
NX - number of nodes in x direction
NY - number of nodes in y direction
NM - number of different materials in model

For blasting problems, the parameter NB is the number of nodes in blastholes.

RECORD 3 - ((X(I,J),I = 1,2),J = 1,NN)

X(1,J) - x coordinate of Jth node
X(2,J) - y coordinate of Jth node

RECORD 4 - (MATE(I),I = 1,NE)

MATE(I) - material number of Ith element

RECORD 5 - ((KODE(I,J),I = 1,2),J = 1,NN)

$$\text{KODE}(I,J) = \begin{cases} 1 & \text{Ith degree of freedom fixed } (=0) \text{ at Jth node} \\ 0 & \text{Ith degree of freedom unknown at Jth node} \end{cases}$$

RECORD 6 - ((P(I,J),I = 1,4),J = 1,NM)

P(1,J) - Young's modulus of Jth material
P(2,J) - Poisson's ratio of Jth material
P(3,J) - density of Jth material
P(4,J) - tensile strength of Jth material

P(4,J) is not used if the job is not a damage analysis.

RECORD 7 - (NBC(I),I = 1,NB)

NBC(I) - node number of Ith node where tractions are specified

RECORD 8 - ((FG(I,J),I = 1,2),J = 1,NN)

FG(2,I) - geometric part of load vector for Ith node (see Section 2.4, Equation 2.15)

If the answer to the last question is yes, the user is then asked which boundaries are silent. Since the outer boundaries of the mesh form a quadrilateral, a prompt is made for each side (Top, Bottom, Right, Left) to which the user responds with Y(y) or N(n). The program then continues with

Is this a damage analysis?

If the answer is yes, a binary output file, DAM.DAT is opened and the logical variable DAMAGE is set to .TRUE. The user is then asked at which nodes results are to be stored

At how many nodes are results to be stored?

Type node(s) at which results are to be stored

Results can be stored at any number of nodes (including 0). The limitation is the amount of available core storage.

If the job is a restart, the user is prompted

Type number of additional time steps

All other necessary data is then read from a binary file REST.DAT which must have been created from a previous run. The parameters of the previous run (e.g., nodes at which results are to be stored) remain the same.

All data are stored in a master array which is partitioned once the control parameters are known. If the total storage in this array would be exceeded by execution of the program, an error message is printed on the screen and execution is terminated. A small (four line) main program sets the size of the master array and calls the subroutine BLAST which is the main driver.

A subroutine FVEC is used to provide the time functions describing the forces in the blast-hole(s) (See Section 2.4). This must be programmed by the user. A convenient way of operating this program is to keep the small main program and subroutine FVEC in the file called BLAST.FOR. The remaining subroutines are stored in another file, BSUB.FOR. Changes to storage and time functions may then be easily made. The program is made into an executable module by first compiling BLAST.FOR and BSUB.FOR and then linking them with the DOS linker

LINK BLAST BSUB,,,,

which results in an executable module called BLAST.EXE.

The program produces the following output files:

DVA.DAT - time history file (unit 4) used by PLOT
FE.OUT - ASCII output file (unit 6)
DAM.DAT - damaged elements file (unit 7) used by DDAM
REST.DAT - restart file (unit 8)

The contents of the file DVA.DAT are shown on the next page. The file FE.OUT is used as a readable record of input and output data. If the analysis is for damage, a file DAM.DAT is created which contains information on the state (failed or intact) of each element at the final time step. The restart file is written at the end of each run, overwriting any previous restart files.

Contents of File DVA.DAT

RECORD 1 - NT,TB,DT

NT - number of time steps
TB - starting time for time histories (usually 0)
DT - time step

For each node at which results were stored, the following records are written:

1) NODE - node number
2) u_x - x component of displacement
3) u_y - y component of displacement
4) $\dot{u}_x$ - x component of velocity
5) $\dot{u}_y$ - y component of velocity
6) $\ddot{u}_x$ - x component of acceleration
7) $\ddot{u}_y$ - y component of acceleration

There must be NT values of displacement, velocity and acceleration in each record.

In some cases, underflows will occur during the finite element analysis, resulting in termination of execution. These may be avoided by turning off underflow warnings and treating small numbers as zero. For the Lahey compiler, the subroutine call

```
CALL UNDER0 (.TRUE.)
```

will turn off underflow warnings. This call is in the main program and is the only compiler dependent feature of the finite element processor.

B.5 PLOT

The program PLOT plots time history data stored in the binary file DVA.DAT created by BLAST. The time histories may be displayed on either a screen, a printer or a plotter.

The program is stored in a file called PLOT.FOR and is made into an executable module by first compiling it and then linking it to the library of graphics routines associated with the particular FORTRAN compiler in use. For the Lahey compiler, the LINK statement which forms the executable module is

LINK PLOT, , , LAHEYCGI

which results in an executable module called PLOT.EXE. The correct drivers for the screen, printing and plotting devices must be loaded before executing PLOT.

By pressing the CARRIAGE RETURN (or ENTER) key, PLOT will display the time history of each component of displacement, velocity and acceleration at a node in turn. The peak values of each time history are also displayed. The time histories are first displayed on the screen. The following commands may then be issued:

R (or r) - display time history on printer
L (or l) - display time history on plotter
Q (or q) - stop and return to DOS

When the program finishes plotting on a hard copy device, the screen then becomes the workstation and the following prompt is given:

Press Q(q) to quit or CARRIAGE RETURN to continue

A CARRIAGE RETURN causes the next time history to be displayed on the screen. When the program runs out of data to display, the following prompt is given:

No data left. Press Q(q) to quit or CARRIAGE RETURN to restart

The CARRIAGE RETURN will cause the file DVA.DAT to be rewound and display to start from the beginning.

B.6 DDAM

The program DDAM displays the damage pattern stored in the binary file DAM.DAT created by BLAST-BSUB. The damage pattern may be displayed on a screen, a printer or a plotter. The damaged elements are filled with a hatch pattern.

The program is stored in a file called DDAM.FOR and is made into an executable module by first compiling it and then linking it to the library of graphics routines associated with the particular FORTRAN compiler in use. For the Lahey compiler, the LINK statement which forms the executable module is

LINK DDAM, , , LAHEYCGI

which results in an executable module called DDAM.EXE. The correct drivers for the screen, printing and plotting devices must be loaded before executing DDAM.

Whatever the display device used, the user is prompted for a scale factor, SF, in the range $0 \leq SF \leq 1$ which determines the portion of the display area which will be used to display the mesh. Scale factors less than 1.0 are often useful on hard copy devices. The mesh is first displayed on the screen. The following commands may then be issued:

R (or r) - display damage on printer
L (or l) - display damage on plotter
Q (or q) - stop and return to DOS

APPENDIX C

Program Listing

The source listing of the finite element processor is shown on the following pages. The main driver, subroutines FVEC and BLAST are shown first. All remaining subroutines are then shown in alphabetical order. In the interest of brevity, listings of the pre and post-processors are not given. All source code is provided on the diskettes provided with this report.

```
      DIMENSION A(40000)
C
C MAIN DRIVER
C
      MAX = 40000
      CALL BLAST (A,MAX)
      END

      SUBROUTINE FVEC (F,KODE,NBC,FG,T)
C-----------------------------------------------------------------
C
C COMPUTE LOAD VECTOR AT TIME T
C
C-----------------------------------------------------------------
      COMMON /CNTRL/ NB,NE,NK,NN,NX,NY,NMAT,NN2,NEQ
      DIMENSION F(NEQ),KODE(2,NN),NBC(NB),FG(2,NN)
C BLAST MODEL 1
      DATA A,BETA,OMEGA/1.3274E8,9.669E3,3.142E3/
C BLAST MODEL 2
C     DATA A,BETA,OMEGA/2.995E10,9.669E3,3.142E3/
C KIDD CREEK EQUIVALENT CAVITY MODEL
C     DATA A,BETA,OMEGA/1.149E9, 9.669E3,3.142E3/
      PARAMETER (PI=3.141593)
C
      CALL ZERO (F,NEQ)
C
C TIME FUNCTION FOR ONE DIMENSIONAL LOADED HALF SPACE
      P    = 200.E6*SIN(10.*PI*T)
C
C     P = 0.0
C     IF (T .LT. 0.0 ) RETURN
C     P = A*EXP(-BETA*T)*SIN(OMEGA*T)
      DO 10 I = 1,NB
        NODE = NBC(I)
        IF (KODE(1,NODE) .NE. 0) F(KODE(1,NODE)) = P  *FG(1,NODE)
        IF (KODE(2,NODE) .NE. 0) F(KODE(2,NODE)) = P  *FG(2,NODE)
   10 CONTINUE
C
      END
```

```
      SUBROUTINE BLAST (A,MAX)
C-----------------------------------------------------------------------
C
C FINITE ELEMENT PROGRAM FOR WAVE PROPAGATION
C
C-----------------------------------------------------------------------
      DIMENSION A(MAX)
      CHARACTER*80 TITLE
      CHARACTER*1 ANS
      COMMON /CNTRL/ NB,NE,NK,NM,NN,NX,NY,NN2,NEQ
      COMMON /TPARM/ DT,NTS,NT
      COMMON /SHHHH/ SB(4)
      COMMON /DPARM/ DAMAGE
      LOGICAL SB,DAMAGE
C
C ZERO MASTER ARRAY
C
      CALL ZERO (A,MAX)
C
C TURN OFF UNDERFLOW WARNINGS (COMPILER DEPENDENT)
C
      CALL UNDER0(.TRUE.)
C
C OPEN FILES
C
C     3 - INPUT DATA FILE
C     4 - OUTPUT TIME HISTORY DATA FILE
C     6 - ASCII OUTPUT FILE
C     7 - DAMAGE FILE (IF DAMAGE ANALYSIS, SEE BELOW)
C     8 - RESTART FILE
C
      OPEN (3,FILE='MESH.DAT',FORM='UNFORMATTED')
      OPEN (4,FILE='DVA.DAT',FORM='UNFORMATTED')
      OPEN (6,FILE='FE.OUT',CARRIAGE CONTROL='FORTRAN')
      OPEN (8,FILE='REST.DAT',FORM='UNFORMATTED')
C
      REWIND 3
      READ (3) TITLE
      PRINT *, TITLE
      READ (3) NB,NE,NN,NX,NY,NM
C
      WRITE (6,600) TITLE
  600 FORMAT ('1',A80)
      WRITE (6,601) NE,NN,NX,NY,NM,NB
  601 FORMAT (/' Control Parameters'//
     1         ' ',5X,'Number of elements                ',I5/
     2         ' ',5X,'Number of nodes                   ',I5/
     3         ' ',5X,'Number of nodes in x direction ',I5/
     4         ' ',5X,'Number of nodes in y direction ',I5/
     5         ' ',5X,'Number of materials               ',I5/
     6         ' ',5X,'Number of loaded nodes            ',I5)
```

```
C
      NN2 = 2*NN
      N2 = 1 + NN2
      N3 = N2 + NN2
      N4 = N3 + NE
      N5 = N4 + 4*NM
      N6 = N5 + NB
C                  X KODE  MATE    P   NBC    FG
      CALL INPUT (A,A(N2),A(N3),A(N4),A(N5),A(N6))
C
C NUMBER EQUATIONS
C
      CALL EQNO (A(N2))
      WRITE (6,602) NEQ
  602 FORMAT (/' Number of equations ',I5)
C
C SET POINTERS FOR MASTER ARRAY
C
      N7 = N6 + NN2
      N8 = N7 + NEQ
      N9 = N8 + NEQ
      N10 = N9 + NEQ
      N11 = N10 + NEQ
      N12 = N11 + NEQ
      N13 = N12 + NEQ
C
      NTS = 0
      PRINT *, 'Is this a restart? '
      READ *, ANS
      IF (ANS .EQ. 'Y' .OR. ANS .EQ. 'y') THEN
        READ (8) DT,NTS,NK,DAMAGE
        PRINT *, 'Type number of additional time steps '
        READ *, NTADD
        NT = NTS + NTADD
        WRITE (6,603) NTS,NTADD,DT
  603   FORMAT (/' Restart run'//
     1          ' ',5X,'Old number of time steps ',I5/
     2          ' ',5X,'Additional time steps    ',I5/
     3          ' ',5X,'Time step                ',1PE10.3)
        IF (DAMAGE) THEN
          OPEN (7,FILE='DAM.DAT',FORM='UNFORMATTED')
        ENDIF
        N14 = N13 + NK
        N15 = N14 + NT*2*NK
        N16 = N15 + NT*2*NK
        N17 = N16 + NT*2*NK
        NEND = N17 + NE
        IF (NEND-1 .GT. MAX) CALL ERROR (1,NEND-1-MAX)
        CALL REST (A(N7),A(N8),A(N10),A(N11),A(N12),A(N13),A(N14),
     1             A(N15),A(N16),A(N17))
```

```
      ELSE
        PRINT *, 'Type number of time steps '
        READ *, NT
        PRINT *, 'Type time step '
        READ *, DT
        WRITE (6,604) NT,DT
  604   FORMAT (/' Number of time steps ',I5/
     1            ' Time step            ',1PE10.3)
C
        DO 10 I = 1,4
          SB(I) = .FALSE.
   10   CONTINUE
        PRINT *, 'Are there any silent boundaries in the model? '
        READ *, ANS
        IF (ANS .EQ. 'Y' .OR. ANS .EQ. 'y') THEN
          PRINT *, 'Top side? '
          READ *, ANS
          IF (ANS .EQ. 'Y' .OR. ANS .EQ. 'y') SB(1) = .TRUE.
          PRINT *, 'Left side? '
          READ *, ANS
          IF (ANS .EQ. 'Y' .OR. ANS .EQ. 'y') SB(2) = .TRUE.
          PRINT *, 'Bottom side? '
          READ *, ANS
          IF (ANS .EQ. 'Y' .OR. ANS .EQ. 'y') SB(3) = .TRUE.
          PRINT *, 'Right side? '
          READ *, ANS
          IF (ANS .EQ. 'Y' .OR. ANS .EQ. 'y') SB(4) = .TRUE.
        ENDIF
C
        DAMAGE = .FALSE.
        PRINT *, 'Is this a damage analysis? '
        READ *, ANS
        IF (ANS .EQ. 'Y' .OR. ANS .EQ. 'y') THEN
          DAMAGE = .TRUE.
          OPEN (7,FILE='DAM.DAT',FORM='UNFORMATTED')
        ENDIF
        CALL KEEP (A(N13))
        N14 = N13 + NK
        N15 = N14 + NT*2*NK
        N16 = N15 + NT*2*NK
        N17 = N16 + NT*2*NK
        NEND = N17 + NE
        IF (NEND-1 .GT. MAX) CALL ERROR (1,NEND-1-MAX)
C
C COMPUTE MASS MATRIX
C                    X KODE  MATE    P       M
        CALL AMASS (A,A(N2),A(N3),A(N4),A(N7))
```

```
C
C SET UP SILENT BOUNDARIES
C                                MATE    P     C
        CALL ADAMP (A,A(N2),A(N3),A(N4),A(N8))
      ENDIF
C
C STEP THROUGH TIME
C                    X KODE  MATE    P     NBC     FG      M      C      F
      CALL SOLVIT (A,A(N2),A(N3),A(N4),A(N5),A(N6),A(N7),A(N8),A(N9),
C         DIS    VEL     ACC     KN      DK      VK      AK      IDAM
     1 A(N10),A(N11),A(N12),A(N13),A(N14),A(N15),A(N16),A(N17))
C
C WRITE OUTPUT FILES
C
      CALL OUTPUT (A(N7),A(N8),A(N10),A(N11),A(N12),A(N13),A(N14),
     1 A(N15),A(N16),A(N17))
C
      END
```

```
      SUBROUTINE ADAMP (X,KODE,MATE,P,C)
C-----------------------------------------------------------------------
C
C ASSEMBLE LUMPED DAMPING MATRIX FOR SILENT BOUNDARIES
C
C-----------------------------------------------------------------------
      COMMON /CNTRL/ NB,NE,NK,NM,NN,NX,NY,NN2,NEQ
      COMMON /SHHHH/ SB(4)
      LOGICAL SB
      DIMENSION X(NN2),KODE(NN2),MATE(NE),P(4,NM),C(NEQ)
      DIMENSION XL(8),KL(8)
C
      CALL ZERO (C,NEQ)
C
C SIDE 1
C
      IF (SB(1)) THEN
        DO 10 N = NE-NX+2,NE
          CALL TEDIUM (N,X,XL,KODE,KL,MATE,P,VP,VS,RHO)
          F = (XL(3)-XL(1))*0.5*RHO
          VPF = VP*F
          VSF = VS*F
          IF (KL(1) .NE. 0) C(KL(1)) = C(KL(1)) + VSF
          IF (KL(2) .NE. 0) C(KL(2)) = C(KL(2)) + VPF
          IF (KL(3) .NE. 0) C(KL(3)) = C(KL(3)) + VSF
          IF (KL(4) .NE. 0) C(KL(4)) = C(KL(4)) + VPF
   10   CONTINUE
      ENDIF
C
C SIDE 2
C
      IF (SB(2)) THEN
        DO 20 N = 1,NE-NX+2,NX-1
          CALL TEDIUM (N,X,XL,KODE,KL,MATE,P,VP,VS,RHO)
          F = (XL(4)-XL(6))*0.5*RHO
          VPF = VP*F
          VSF = VS*F
          IF (KL(3) .NE. 0) C(KL(3)) = C(KL(3)) + VPF
          IF (KL(4) .NE. 0) C(KL(4)) = C(KL(4)) + VSF
          IF (KL(5) .NE. 0) C(KL(5)) = C(KL(5)) + VPF
          IF (KL(6) .NE. 0) C(KL(6)) = C(KL(6)) + VSF
   20   CONTINUE
      ENDIF
```

```
C
C SIDE 3
C
      IF (SB(3)) THEN
        DO 30 N = 1,NX-1
          CALL TEDIUM (N,X,XL,KODE,KL,MATE,P,VP,VS,RHO)
          F = (XL(7)-XL(5))*0.5*RHO
          VPF = VP*F
          VSF = VS*F
          IF (KL(5) .NE. 0) C(KL(5)) = C(KL(5)) + VSF
          IF (KL(6) .NE. 0) C(KL(6)) = C(KL(6)) + VPF
          IF (KL(7) .NE. 0) C(KL(7)) = C(KL(7)) + VSF
          IF (KL(8) .NE. 0) C(KL(8)) = C(KL(8)) + VPF
   30   CONTINUE
      ENDIF
C
C SIDE 4
C
      IF (SB(4)) THEN
        DO 40 N = NX-1,NE,NX-1
          CALL TEDIUM (N,X,XL,KODE,KL,MATE,P,VP,VS,RHO)
          F = (XL(2)-XL(8))*0.5*RHO
          VPF = VP*F
          VSF = VS*F
          IF (KL(7) .NE. 0) C(KL(7)) = C(KL(7)) + VPF
          IF (KL(8) .NE. 0) C(KL(8)) = C(KL(8)) + VSF
          IF (KL(1) .NE. 0) C(KL(1)) = C(KL(1)) + VPF
          IF (KL(2) .NE. 0) C(KL(2)) = C(KL(2)) + VSF
   40   CONTINUE
      ENDIF
C
      END
```

```
      SUBROUTINE AMASS (X,KODE,MATE,P,LM)
C-----------------------------------------------------------------------
C
C ASSEMBLE LUMPED MASS MATRIX
C
C-----------------------------------------------------------------------
      COMMON /CNTRL/ NB,NE,NK,NM,NN,NX,NY,NN2,NEQ
      DIMENSION X(NN2),KODE(NN2),MATE(NE),P(4,NM),LM(NEQ)
      DIMENSION XL(8),KL(8),NEC(4),LML(8)
      REAL LM,LML
C
      CALL ZERO (LM,NEQ)
      DO 20 N = 1,NE
        CALL FIJ (N,IN,JN)
        CALL GETNEC (IN,JN,NEC)
        CALL GTOL (KODE,KL,NEC)
        CALL GTOL (X,XL,NEC)
        RHO = P(3,MATE(N))
        CALL LMASSM (N,XL,RHO,LML)
        DO 10 I = 1,8
          IF (KL(I) .NE. 0) THEN
            LM(KL(I)) = LM(KL(I)) + LML(I)
          ENDIF
   10   CONTINUE
   20 CONTINUE
C
      END
```

```
      SUBROUTINE DM (XC,B,H,DETJ,R,S,NEL)
C-----------------------------------------------------------------
C
C COMPUTE DERIVATIVES AND JACOBIAN AT INTEGRATION POINT (R,S)
C
C-----------------------------------------------------------------
      DIMENSION XC(2,4),B(2,4),P(2,4),XJ(2,2),XJI(2,2),H(4)
C
      RP = 1.0 + R
      RM = 1.0 - R
      SP = 1.0 + S
      SM = 1.0 - S
C
      P(1,1) = 0.25*SP
      P(1,2) = -P(1,1)
      P(1,3) = -0.25*SM
      P(1,4) = -P(1,3)
C
      P(2,1) = 0.25*RP
      P(2,2) = 0.25*RM
      P(2,3) = -P(2,2)
      P(2,4) = -P(2,1)
C
      H(1) = 0.25*RP*SP
      H(2) = 0.25*RM*SP
      H(3) = 0.25*RM*SM
      H(4) = 0.25*RP*SM
C
C COMPUTE JACOBIAN
C
      DO 20 I = 1,2
      DO 20 J = 1,2
        SUM = 0.0
      DO 10 K = 1,4
        SUM = SUM + P(I,K)*XC(J,K)
  10  CONTINUE
        XJ(I,J) = SUM
  20  CONTINUE
C
C COMPUTE DETERMINANT AND INVERSE OF JACOBIAN
C
      DETJ = XJ(1,1)*XJ(2,2) - XJ(1,2)*XJ(2,1)
      IF (DETJ .LE. 1.E-6) THEN
        PRINT *, 'Small or negative determinant for element ',NEL
        PRINT *,XC
      ENDIF
```

```
C
      XJI(1,1) = XJ(2,2)/DETJ
      XJI(1,2) = -XJ(1,2)/DETJ
      XJI(2,1) = -XJ(2,1)/DETJ
      XJI(2,2) = XJ(1,1)/DETJ
C
C EVALUATE GLOBAL DERIVATIVE OPERATOR B
C
      DO 50 I = 1,2
      DO 40 J = 1,4
        SUM = 0.0
      DO 30 K = 1,2
        SUM = SUM + XJI(I,K)*P(K,J)
  30  CONTINUE
        B(I,J) = SUM
  40  CONTINUE
  50  CONTINUE
C
      END

      SUBROUTINE EQNO (KODE)
C-----------------------------------------------------------------------
C
C NUMBER EQUATIONS
C
C-----------------------------------------------------------------------
      COMMON /CNTRL/ NB,NE,NK,NM,NN,NX,NY,NN2,NEQ
      DIMENSION KODE(NN2)
C
      NEQ = 0
      DO 10 I = 1,NN2
        IF (KODE(I) .EQ. 0) THEN
          NEQ = NEQ + 1
          KODE(I) = NEQ
        ELSE
          KODE(I) = 0
        ENDIF
   10 CONTINUE
C
      END
```

```
      SUBROUTINE ERROR (IE,N)
C-----------------------------------------------------------------
C
C FATAL ERRORS
C
C-----------------------------------------------------------------
      IF (IE .LE. 0) RETURN
      GO TO (1,2,3), IE
C
    1 PRINT *, 'Storage exceeded by ',N
      STOP
C
    2 RETURN
C
    3 RETURN
C
      END

      SUBROUTINE ESTR (B,UL,EPS)
C-----------------------------------------------------------------
C
C COMPUTE STRAIN OR STRAIN RATE IN ELEMENT
C
C-----------------------------------------------------------------
      DIMENSION B(2,4),UL(2,4),EPS(3)
C
      CALL ZERO (EPS,3)
      DO 10 J = 1,4
        EPS(1) = EPS(1) + B(1,J)*UL(1,J)
        EPS(2) = EPS(2) + B(2,J)*UL(2,J)
        EPS(3) = EPS(3) + B(1,J)*UL(2,J) + B(2,J)*UL(1,J)
   10 CONTINUE
C
      END
```

```
      SUBROUTINE ETOL (AG,AL,KL)
C-----------------------------------------------------------------
C
C GET LOCAL ATTRIBUTE ARRAY AL FROM EQUATION ARRAY AG
C
C-----------------------------------------------------------------
      COMMON /CNTRL/ NB,NE,NK,NM,NN,NX,NY,NN2,NEQ
      DIMENSION AG(NEQ),AL(8),KL(8)
C
      CALL ZERO (AL,8)
      DO 10 I = 1,8
        KLI = KL(I)
        IF (KLI .NE. 0) THEN
          AL(I) = AG(KLI)
        ENDIF
   10 CONTINUE
C
      END

      SUBROUTINE FIJ (N,I,J)
C-----------------------------------------------------------------
C
C FIND I J OF ELEMENT N
C
C-----------------------------------------------------------------
      COMMON /CNTRL/ NB,NE,NK,NM,NN.NX,NY,NN2,NEQ
C
      I = MOD(N,NX-1)
      IF (I .EQ. 0) I = NX-1
      J = (N-I)/(NX-1) + 1
C
      END

      SUBROUTINE GETNEC (IN,JN,NEC)
C-----------------------------------------------------------------
C
C GET ELEMENT CONNECTIONS NEC FROM I J OF ELEMENT
C
C-----------------------------------------------------------------
      COMMON /CNTRL/ NB,NE,NK,NM,NN,NX,NY,NN2,NEQ
      DIMENSION NEC(4)
C
      NEC(3) = IN + (JN-1)*NX
      NEC(2) = NEC(3) + NX
      NEC(4) = NEC(3) + 1
      NEC(1) = NEC(4) + NX
C
      END
```

```
      SUBROUTINE GTOL (AG,AL,NEC)
C-----------------------------------------------------------------
C
C GET LOCAL ATTRIBUTE ARRAY AL FROM GLOBAL ARRAY AG
C
C-----------------------------------------------------------------
      COMMON /CNTRL/ NB,NE,NK,NM,NN,NX,NY,NN2,NEQ
      DIMENSION AG(2,NN),AL(2,4),NEC(4)
C
      DO 20 J = 1,4
        NECJ = NEC(J)
        DO 10 I = 1,2
          AL(I,J) = AG(I,NECJ)
   10   CONTINUE
   20 CONTINUE
C
      END

      SUBROUTINE INPUT (X,KODE,MATE,P,NBC,FG)
C-----------------------------------------------------------------
C
C INPUT MODEL GEOMETRY, ETC FROM EXTERNAL DATA FILE
C
C-----------------------------------------------------------------
      COMMON /CNTRL/ NB,NE,NK,NM,NN,NX,NY,NN2,NEQ
      DIMENSION X(2,NN),KODE(2,NN),MATE(NE),P(4,NM)
      DIMENSION NBC(NB),FG(2,NN)
C
      READ (3) ((X(I,N),I = 1,2),N = 1,NN)
      READ (3) (MATE(I),I = 1.NE)
      READ (3) ((KODE(I,N),I = 1,2),N = 1,NN)
      READ (3) ((P(I,J),I = 1,4),J = 1,NM)
      READ (3) (NBC(I),I = 1,NB)
      READ (3) ((FG(I,N),I = 1,2),N = 1,NN)
C
      WRITE (6,605)
  605 FORMAT (/' Material Properties'//
     1 ' ',5X,' #',14X,'E',4X,'PR',12X,'RHO',11X,'SIGT',13X,'CP'/)
      DO 10  J = 1,NM
        E = P(1,J)
        PR = P(2,J)
        RHO = P(3,J)
        SIGT = P(4,J)
        CP = SQRT(E*(1.-PR)/(1.+PR)/(1.-2.*PR)/RHO)
        WRITE (6,610) J,E,PR,RHO,SIGT,CP
   10 CONTINUE
  610 FORMAT (' ',5X,I2,E15.4,F6.2,3E15.4)
C
      WRITE (6,615)
  615 FORMAT (/' The following nodes are loaded'/)
      WRITE (6,620) (NBC(I),I = 1,NB)
  620 FORMAT (' ',10I5)
C
      END
```

```
      SUBROUTINE KEEP (KN)
C-----------------------------------------------------------------------
C
C READ NODES AT WHICH RESULTS ARE TO BE STORED
C
C-----------------------------------------------------------------------
      DIMENSION KN(*)
      COMMON /CNTRL/ NB,NE,NK,NM,NN,NX,NY,NN2,NEQ
C
      PRINT *, 'At how many nodes are results to be stored? '
      READ *, NK
      IF (NK .EQ. 0) RETURN
      PRINT *, 'Type node(s) at which results are to be stored'
      PRINT *,
      READ *, (KN(I),I = 1,NK)
C
      END

      SUBROUTINE LMASSM (NEL,XC,RHO,AM)
C-----------------------------------------------------------------------
C
C COMPUTE LUMPED MASS MATRIX
C
C-----------------------------------------------------------------------
      DIMENSION XC(2,4),AM(8)
      DIMENSION H(4),B(2,4),R(4),S(4)
      DATA R/1.0,-1.0,-1.0,1.0/
      DATA S/1.0,1.0,-1.0,-1.0/
C
      CALL ZERO (AM,8)
      M = 1
      DO 10 I = 1,4
        CALL DM (XC,B,H,DETJ,R(I),S(I),NEL)
        AM(M  ) = AM(M) + RHO*DETJ
        AM(M+1) = AM(M)
        M = M + 2
   10 CONTINUE
C
      END
```

```
      SUBROUTINE OUTPUT (M,C,DIS,VEL,ACC,KN,DK,VK,AK,IDAM)
C-----------------------------------------------------------------------
C
C OUTPUT TIME HISTORY DATA AT SPECIFIED NODES
C
C-----------------------------------------------------------------------
      COMMON /CNTRL/ NB,NE,NK,NM,NN,NX,NY,NN2,NEQ
      COMMON /TPARM/ DT,NTS,NT
      COMMON /DPARM/ DAMAGE
      REAL M
      DIMENSION M(NEQ),C(NEQ),DIS(NEQ),VEL(NEQ),ACC(NEQ)
      DIMENSION KN(*),DK(NT,2,*),VK(NT,2,*),AK(NT,2,*),IDAM(NE)
      LOGICAL DAMAGE
C
      IF (NK .NE. 0) THEN
        REWIND 4
        TB = 0.0
        WRITE (4) NT,TB,DT
        DO 40 K = 1,NK
          WRITE (4) KN(K)
          DO 10 J = 1,2
            WRITE (4) (DK(I,J,K),I = 1,NT)
   10     CONTINUE
          DO 20 J = 1,2
            WRITE (4) (VK(I,J,K),I = 1,NT)
   20     CONTINUE
          DO 30 J = 1,2
            WRITE (4) (AK(I,J,K),I = 1,NT)
   30     CONTINUE
   40   CONTINUE
      ENDIF
C
      IF (DAMAGE) WRITE (7) (IDAM(I),I = 1,NE)
C
C WRITE RESTART FILE
C
      REWIND 8
      WRITE (8) DT,NT,NK,DAMAGE
      WRITE (8) (M(I),I = 1,NEQ)
      WRITE (8) (C(I),I = 1,NEQ)
      WRITE (8) (DIS(I),I = 1,NEQ)
      WRITE (8) (VEL(I),I = 1,NEQ)
      WRITE (8) (ACC(I),I = 1,NEQ)
C
      END
```

```
      SUBROUTINE QVEC (XC,DISL,E,V,IDAM,SIGT,Q,NEL)
C-----------------------------------------------------------------------
C
C COMPUTE INTERNAL FORCE VECTOR Q FOR ELEMENT NEL
C
C PLANE STRAIN FORMULATION
C
C-----------------------------------------------------------------------
      DIMENSION XC(2,4),DISL(2,4),Q(8)
      DIMENSION H(4),B(2,4),XG(2),WG(2),EPS(3)
      COMMON /DPARM/ DAMAGE
      LOGICAL DAMAGE
      DATA XG/-0.57735, 0.57735/
      DATA WG/1.0     , 1.0    /
      DATA NG/2/
C
      CALL ZERO (Q,8)
      IF (IDAM .EQ. 1) RETURN
C
      THREEB = E/(1.-2.*V)
      TWOG = E/(1.+V)
C
      DO 40 M = 1,NG
        RM = XG(M)
        WM = WG(M)
        DO 30 N = 1,NG
          SN = XG(N)
          WN = WG(N)
          CALL DM (XC,B,H,DETJ,RM,SN,NEL)
          CALL ESTR (B,DISL,EPS)
C
          VEPS = EPS(1) + EPS(2)
          SIGMA = THREEB*VEPS
          EXX = EPS(1) - VEPS/3.
          EYY = EPS(2) - VEPS/3.
          EXY = EPS(3)
          SXX = TWOG*EXX
          SYY = TWOG*EYY
          SXY = TWOG*EXY
          SIGXX = SXX + SIGMA/3.
          SIGYY = SYY + SIGMA/3.
          SIGXY = SXY
C
          WT = WM*WN*DETJ
          DO 20 J = 1,4
            Q(J+J-1) = Q(J+J-1) + WT*(B(1,J)*SIGXX + B(2,J)*SIGXY)
            Q(J+J  ) = Q(J+J  ) + WT*(B(1,J)*SIGXY + B(2,J)*SIGYY)
   20     CONTINUE
   30   CONTINUE
   40 CONTINUE
```

```
C
      IF (DAMAGE) THEN
        CALL DM (XC,B,H,DETJ,0.0,0.0,NEL)
        CALL ESTR (B,DISL,EPS)
        SIGMA = THREEB*(EPS(1)+EPS(2))
        IF (SIGMA .GT. SIGT .AND. IDAM .EQ. 0) THEN
          IDAM = 1
        ENDIF
      ENDIF
C
      END

      SUBROUTINE REST (M,C,DIS,VEL,ACC,KN,DK,VK,AK,IDAM)
C-----------------------------------------------------------------------
C
C READ RESTART FILE, OUTPUT TIME HISTORY FILE, AND DAMAGE FILE
C
C-----------------------------------------------------------------------
      COMMON /CNTRL/ NB,NE,NK,NM,NN,NX,NY,NN2,NEQ
      COMMON /TPARM/ DT,NTS,NT
      COMMON /DPARM/ DAMAGE
      LOGICAL DAMAGE
      REAL M
      DIMENSION M(NEQ),C(NEQ),DIS(NEQ),VEL(NEQ),ACC(NEQ)
      DIMENSION KN(*),DK(NT,2,*),VK(NT,2,*),AK(NT,2,*),IDAM(NE)
C
      READ (8) (M(I),I = 1,NEQ)
      READ (8) (C(I),I = 1,NEQ)
      READ (8) (DIS(I),I = 1,NEQ)
      READ (8) (VEL(I),I = 1,NEQ)
      READ (8) (ACC(I),I = 1,NEQ)
C
      IF (NK .NE. 0) THEN
        READ (4) NTOLD,TB,DT
        DO 40 K = 1,NK
          READ (4) KN(K)
          DO 10 J = 1,2
            READ (4) (DK(I,J,K),I = 1,NTOLD)
10        CONTINUE
          DO 20 J = 1,2
            READ (4) (VK(I,J,K),I = 1,NTOLD)
20        CONTINUE
          DO 30 J = 1,2
            READ (4) (AK(I,J,K),I = 1,NTOLD)
30        CONTINUE
40      CONTINUE
      ENDIF
C
      IF (DAMAGE) READ (7) (IDAM(I),I = 1,NE)
C
      END
```

```
      SUBROUTINE SETV (A,N,V)
C-----------------------------------------------------------------------
C
C SET ELEMENTS OF ARRAY A EQUAL TO V
C
C-----------------------------------------------------------------------
      DIMENSION A(*)
C
      DO 10 I = 1,N
        A(I) = V
   10 CONTINUE
C
      END

      SUBROUTINE SOLVIT (X,KODE,MATE,PROP,NBC,FG,M,C,F,DIS,VEL,
     & ACC,KN,DK,VK,AK,IDAM)
C-----------------------------------------------------------------------
C
C SOLVE EQUATIONS OF MOTION USING EXPLICIT
C PREDICTOR-CORRECTOR SCHEME
C
C-----------------------------------------------------------------------
      COMMON /CNTRL/ NB,NE,NK,NM,NN,NX,NY,NN2,NEQ
      COMMON /TPARM/ DT,NTS,NT
      COMMON /DPARM/ DAMAGE
      LOGICAL DAMAGE
      REAL M
      DIMENSION X(NN2),KODE(2,NN),MATE(NE),PROP(4,NM),NBC(NB)
      DIMENSION FG(NN2),M(NEQ),C(NEQ),F(NEQ),DIS(NEQ),VEL(NEQ)
      DIMENSION ACC(NEQ),KN(*),DK(NT,2,*),VK(NT,2,*)
      DIMENSION AK(NT,2,*),IDAM(NE)
      DIMENSION KL(8),XL(8),DISL(8),NEC(4),Q(8)
C
      A1 = 0.5*DT
      A2 = DT*A1
      IF (NK .NE. 0) WRITE (6,600)
  600 FORMAT ('1Time histories (Disp, Vel, Acc)'/)
C
C SOLVE FOR ACCELERATION AT TIME = 0.0
C
      IF (NTS .EQ. 0) THEN
        TIME = 0.0
        CALL FVEC (F,KODE,NBC,FG,TIME)
        DO 10 I = 1,NEQ
          ACC(I) = F(I)/M(I)
   10   CONTINUE
      ENDIF
```

```
C
C STEP THROUGH TIME
C
      DO 80 N = NTS,NT-1
        IF (MOD(N+1,10) .EQ. 0) THEN
          PRINT *, 'TIME STEP ',N+1
        ENDIF
        TIME = (N+1)*DT
C
C UPDATE DISPLACEMENTS TO DIS(TIME)
C
        DO 20 I = 1,NEQ
          DIS(I) = DIS(I) + DT*VEL(I) + A2*ACC(I)
   20   CONTINUE
        CALL FVEC (F,KODE,NBC,FG,TIME)
C
C ADD INTERNAL FORCE VECTOR ELEMENT BY ELEMENT
C
        DO 40 J = 1,NE
C
          NEL = J
          MAT = MATE(NEL)
          IF (IDAM(NEL) .EQ. 0 .AND. MAT .NE. 0) THEN
            CALL FIJ (NEL,IN,JN)
            CALL GETNEC (IN,JN,NEC)
            CALL GTOL (KODE,KL,NEC)
            CALL GTOL (X,XL,NEC)
            E = PROP(1,MAT)
            V = PROP(2,MAT)
            SIGT = PROP(4,MAT)
            CALL ETOL (DIS,DISL,KL)
            CALL QVEC (XL,DISL,E,V,IDAM(NEL),SIGT,Q,NEL)
            DO 30 I = 1,8
              KLI = KL(I)
              IF (KLI .NE. 0) THEN
                F(KLI) = F(KLI) - Q(I)
              ENDIF
   30       CONTINUE
          ENDIF
C
   40   CONTINUE
C
C VELOCITIES AND ACCELERATIONS AT TIME
C
        DO 50 I = 1,NEQ
          P = VEL(I) + A1*ACC(I)
          F(I) = F(I) - C(I)*P
          ACC(I) = F(I)/(M(I) + A1*C(I))
          VEL(I) = P + A1*ACC(I)
   50   CONTINUE
```

```
C
C KEEP THE RESULTS
C
        IF (NK .NE. 0) THEN
        DO 70 K = 1,NK
          KK1 = KODE(1,KN(K))
          KK2 = KODE(2,KN(K))
          IF (KK1 .NE. 0) THEN
            DK(N+1,1,K) = DIS(KK1)
            VK(N+1,1,K) = VEL(KK1)
            AK(N+1,1,K) = ACC(KK1)
            WRITE(6,*) TIME,DIS(KK1),VEL(KK1),ACC(KK1)
          ENDIF
          IF (KK2 .NE. 0) THEN
            DK(N+1,2,K) = DIS(KK2)
            VK(N+1,2,K) = VEL(KK2)
            AK(N+1,2,K) = ACC(KK2)
            WRITE(6,*) TIME,DIS(KK2),VEL(KK2),ACC(KK2)
          ENDIF
   70   CONTINUE
        ENDIF
C
   80 CONTINUE
C
      END

      SUBROUTINE TEDIUM (N,X,XL,KODE,KL,MATE,P,VP,VS,RHO)
C-----------------------------------------------------------------------
C
C COMMON AND TEDIOUS CALCULATIONS ASSOCIATED WITH SILENT BOUNDARY
C
C-----------------------------------------------------------------------
      COMMON /CNTRL/ NB,NE,NK,NM,NN,NX,NY,NN2,NEQ
      DIMENSION X(NN2),KODE(NN2),MATE(NE),P(4,NM)
      DIMENSION XL(8),KL(8),NEC(4)
C
      CALL FIJ (N,IN,JN)
      CALL GETNEC (IN,JN,NEC)
      CALL GTOL (X,XL,NEC)
      CALL GTOL (KODE,KL,NEC)
      MAT = MATE(N)
      E = P(1,MAT)
      PR = P(2,MAT)
      RHO = P(3,MAT)
      VP = SQRT(E*(1.-PR)/(1.+PR)/(1.-2.*PR)/RHO)
      VS = VP*SQRT((1.-2.*PR)/2./(1.-PR))
C
      END
```

```
      SUBROUTINE ZERO (A,N)
C-----------------------------------------------------------------------
C
C NO COMMENT
C
C-----------------------------------------------------------------------
      DIMENSION A(*)
      DO 10 I = 1,N
        A(I) = 0.0
   10 CONTINUE
C
      END
```